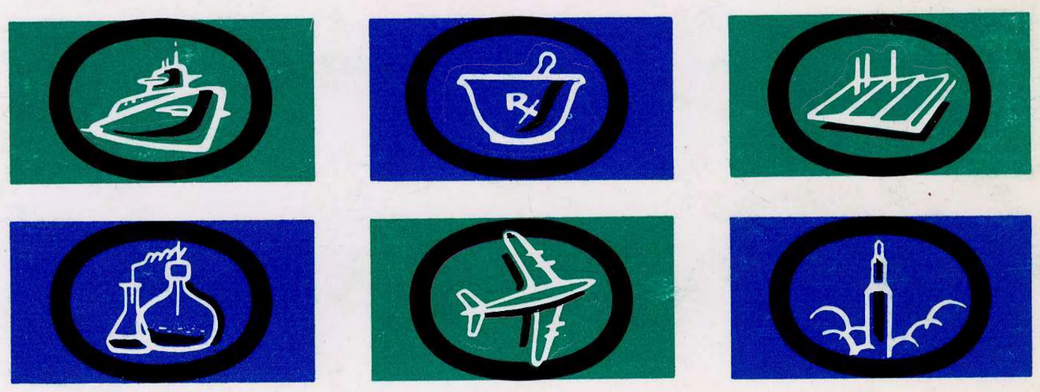
R. LE SAUSSUR
OBSOLETE

REFERENCE MANUAL FOR THE

SEL 840

GENERAL PURPOSE COMPUTER



SEL®

systems engineering laboratories, incorporated

August 1965

REFERENCE MANUAL FOR THE

SEL 840

GENERAL PURPOSE COMPUTER

SEL SYSTEMS ENGINEERING LABORATORIES, INCORPORATED

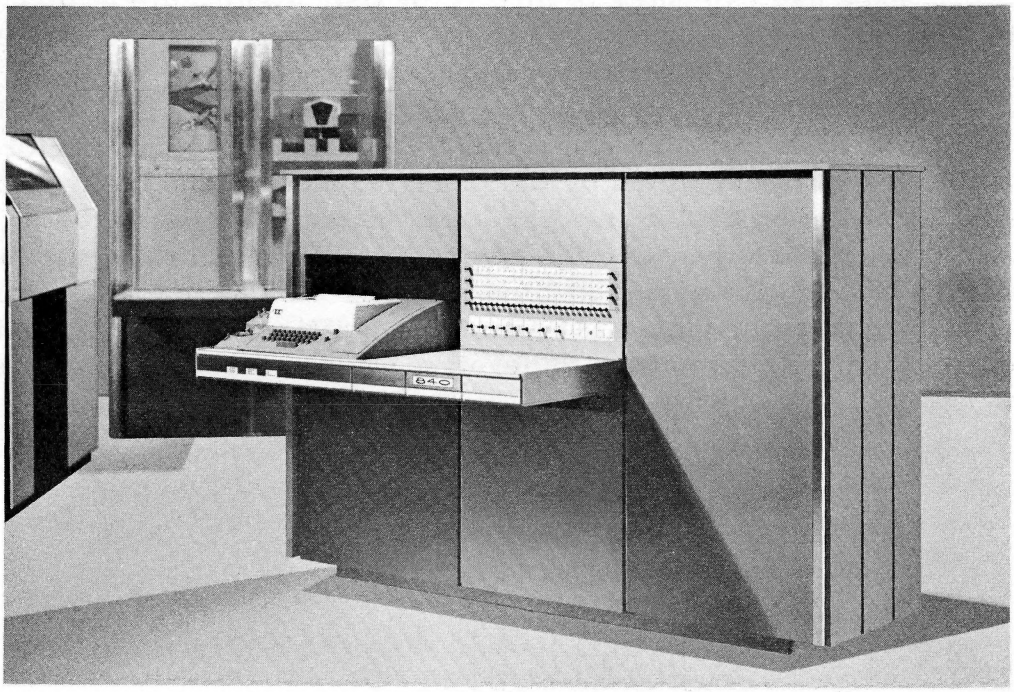
P. O. BOX 9148 • FORT LAUDERDALE, FLA. 33310 • AREA CODE 305 587-2900

SEL 840 REFERENCE MANUAL

TABLE OF CONTENTS

SECTION		PAGE
SECTION I	SEL 840 COMPUTER	1 - 1
	COMPUTER ORGANIZATION	1 - 2
	Memory Unit	1 - 3
	Control Unit	1 - 3
	Arithmetic Unit	1 - 4
	Input/Output Unit	1 - 4
	FUNCTIONAL DESCRIPTION	1 - 7
	Memory Unit	1 - 7
	Control Unit	1 - 7
	Arithmetic Unit	1 - 8
	Input/Output Unit	1 - 11
	Extended Arithmetic Unit (EAU)	1 - 12
	PROGRAMMING THE SEL 840 COMPUTER	1 - 13
	Memory Reference Instructions	1 - 13
	Augmented Instructions	1 - 15
SECTION II	SEL 840 MACHINE LANGUAGE INSTRUCTIONS.	2 - 1
	Load/Store Instructions	2 - 1
	Arithmetic Instructions	2 - 10
	Branch/Skip Instructions	2 - 18
	Logical Instructions	2 - 32
	Register Change Instructions	2 - 39
	Shift Instructions	2 - 46
	Control Instructions	2 - 56
	Input/Output Instructions	2 - 62
	Extended Arithmetic Unit Instructions	2 - 78
SECTION III	SEL 840 SOFTWARE SYSTEM	3 - 1
	840 Assembly Program - MNEMBLER	3 - 2
	SEL 840 Loader	3 - 2
	SEL 840 FORTRAN IV	3 - 3
	SEL 840 Debug	3 - 3
	SEL 840 UPDATE	3 - 4
	SEL 840 Library Package	3 - 4

SECTION I SEL 840 COMPUTER

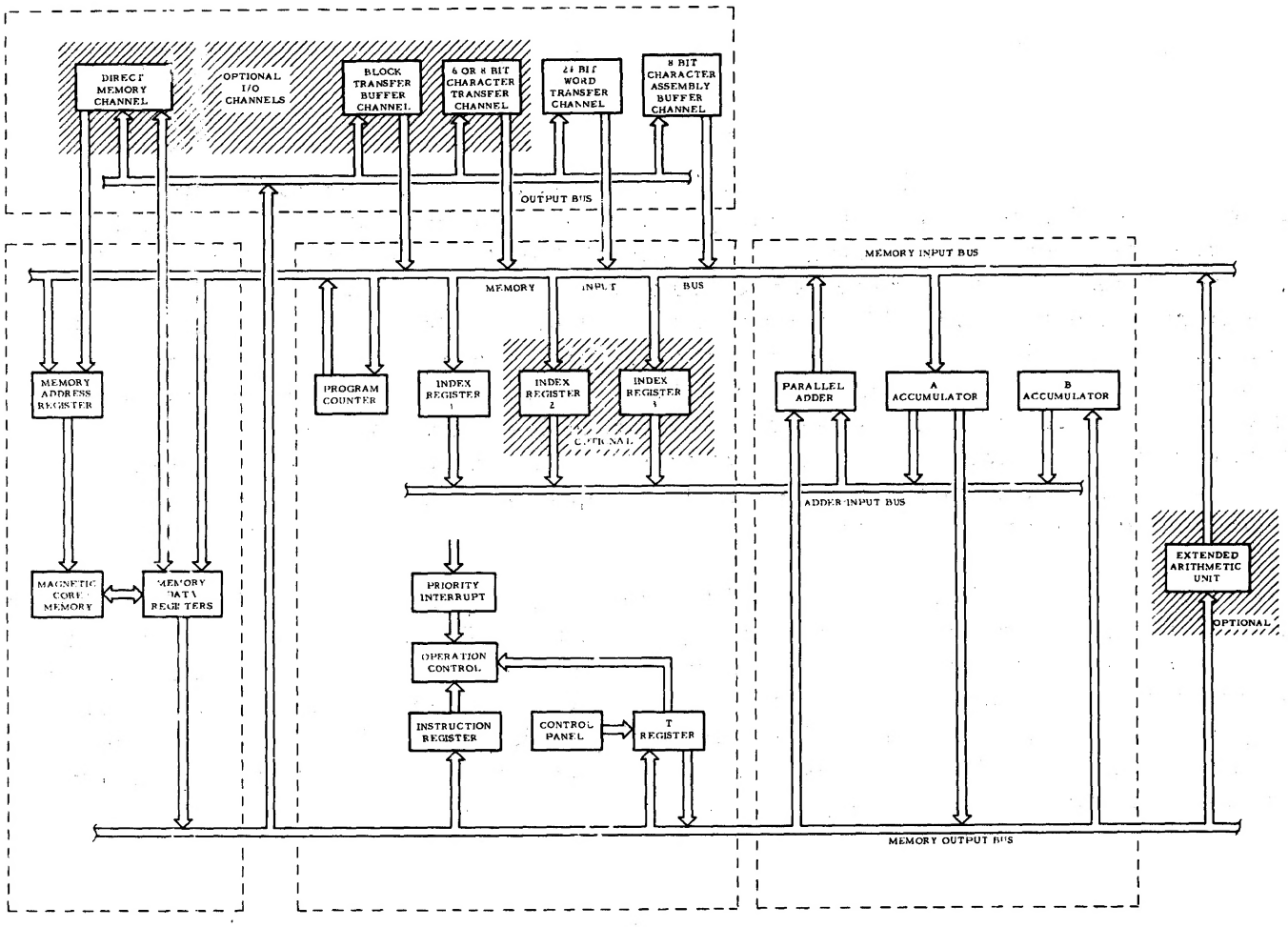


The SEL 840 Scientific Computer is a stored program 24-bit binary parallel computer. The storage element of the computer is a magnetic core memory capable of storing a minimum of 2,048 words and a maximum of 32,768 words. The time required to read or write a word from or to the memory (memory cycle time) is 1.75 microseconds. The SEL 840 has a repertoire of 91 separate instructions, most of which require only one or two memory cycles to be completely executed. The basic machine includes two input/output channels - one for 8-bit character transfers and one for 24-bit word transfers - each of which may be connected to 16 different input/output units. The basic system also includes a hardware index register to allow rapid transfer of data words to and from successive memory locations.

The basic SEL 840 is equipped with a Teletype ASR-33 keyboard/printer that contains a paper tape punch and reader. To this basic input/output configuration may be added magnetic tape units, high-speed line printers, high-speed paper tape readers and punches, card readers and punches, X-Y plotters, CRT displays, magnetic drums and discs and other peripheral devices such as coders, multiplexers, etc. The basic machine may be expanded to a total of 8 input/output channels, up to three index registers and an extended arithmetic unit.

COMPUTER ORGANIZATION

The 840 computer is formed by four major units - memory, control, arithmetic and input/output. The memory stores the instruction words which define the operation of the computer and the data words on which the computer operates. The control unit calls up the instruction words, decodes them and issues commands to operate the computer. The arithmetic unit performs computation with data words supplied by the input/output unit and the memory unit under the direction of the control unit. The input/output unit transmits data words, commands and status reports between the computer and peripheral equipment. The computer operates on, and from, 24-bit binary words which are transmitted in parallel between the computer units. Arithmetic operations are performed using two's complement binary arithmetic with negative words stored in the two's complement form.



SEL 840 Block Diagram

Each of the four major units of the computer is composed of several interconnected elements which are described below.

Memory Unit

Magnetic Core Memory

The memory device is formed by random access, coincident current core stack modules containing 2048, 4096 or 8192 storage locations. Up to eight 4096 or four 8192 modules may be used to provide 32,768 24-bit word storage cells. The memory operates on clear/write and read/restore cycles to retain stored words until deliberately destroyed or replaced. The memory retains stored data even when power is removed and reapplied.

Memory Input and Output Registers

Two 24-bit flip-flop registers which serve as temporary storage for words to be written into or read from the magnetic core memory.

Memory Address Register

A 12-bit flip-flop register which stores memory addresses generated by the control unit. The address is used to gate the read and write coincident currents to the unique group of magnetic cores specified by the address. A 3-bit module selection register in the control unit completes the addressing of the maximum of 32,768 storage locations.

Control Unit

Instruction Register

An 11-bit flip-flop register connected to decoding matrices which provide gating sequences for the various instructions. The instruction register also connects to a 6-stage shift counter preset by the shift instruction words and counted down as the shifts occur.

Decode Matrix

A collection of circuits connected to the instruction register to convert the instruction word operation codes into signals that open communication paths between other elements of the computer.

Priority Interrupt

A group of circuits which automatically switch the program to special program sequences upon receipt of an interrupt signal from peripheral or internal sources. When the special sequence is complete, the program returns to the normal sequence.

Index Registers

One, two or three 15-bit flip-flop registers which may be loaded, stored and incremented by instruction. These registers are individually addressable by instruction word control bits to add their contents to the instruction word operand address. These registers thus allow modification of the operand address during the instruction execution cycle without changing the stored instruction itself and requiring no extra program time. One index register is supplied in the basic configuration.

Program Counter

A 15-bit binary counter that supplies the instruction word addresses to the memory at the start of each instruction cycle. The counter is normally advanced one count each cycle, but may be advanced by two or three counts under special conditions. The counter may be preset to any higher or lower count by a branch instruction.

T Register

A 24-bit flip-flop register that holds one of the operands during arithmetic processes. The T register also holds the address portion of instruction words during the instruction cycles.

Arithmetic Unit

Adder

A 24-bit fully-parallel binary adder with inputs from the T register, A and B accumulators and index register(s). The outputs of the adder are applied to the adder output bus, which provides inputs to the A accumulator, the program counter, index register(s) and the memory.

A Accumulator

A 24-bit flip-flop register with parallel and serial capability that serves as the main arithmetic register and holds the results of all arithmetic operations.

B Accumulator

A 24-bit flip-flop register with parallel and serial capability that serves as an extension of the A accumulator for multiplication and division.

The contents of both the A and B accumulators may be shifted right or left, separately or together. These shifts may either be logical (all bits including sign) or arithmetic (excluding the sign bit).

Input/Output Unit

Character Assembly Channel

Provides for the automatic conversion of 24-bit words to three 8-bit or four 6-bit characters and the reverse for transfers between the computer and character-oriented I/O devices.

Word Channel

Provides for the parallel transfer of 24 bits of input or output data during each cycle. This channel serves to communicate with word-oriented I/O devices.

* Character Channel

Provides for the transfer of 6, 8 or 12 bits of input or output data during each cycle. The conversion of the 24-bit computer words to and from the 6, 8 or 12-bit input/output characters is accomplished by the program.

* Block Transfer Channel

Provides for the automatic transfer of blocks of input or output data. The computer supplies the starting memory address and block-length count to the channel control counter and initiates the transfer. The transfers continue at rates up to 570 KC until transfer count has been reduced to zero.

* Direct Memory Channel

Provides for completely separate operation of an I/O device with the computer memory. May be operated as a word, character, block transfer or multiplexed block transfer channel.

Up to a maximum of eight I/O channels may be used with the SEL 840 with each channel capable of connecting 16 separate I/O units to the computer. All channels except for the direct memory channels connect to the computer bus structure.

* Extended Arithmetic Unit

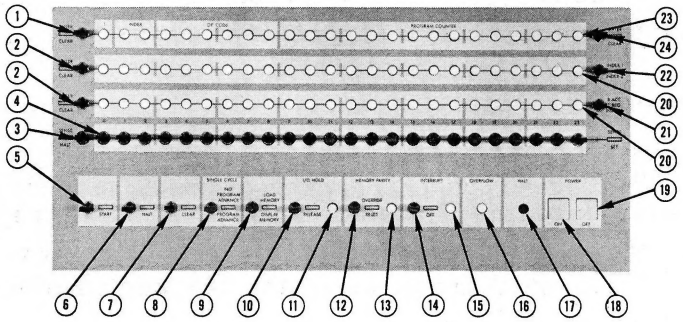
The extended arithmetic unit is an optional device containing complete hardware for single precision, double precision, fixed point and floating point arithmetic. The EAU contains two 48-bit accumulators and operates independently of the main frame arithmetic unit, but sharing control and memory with the remainder of the main frame.

* Optional Units

Control Panel

The gross computer control - ON, OFF, START, STOP - is performed by the switches on the control console (see below). In addition to this, the program may be monitored and altered through the use of indicators and switches located on the console. A group of 24 switches allows entry of data, when requested by the computer through a "load control switches" instruction, and branching of the program, through the use of a "switch not set" instruction. The indicators may be switched to connect to each major register so that the contents of these registers may be observed at any time. The registers may be loaded with bits through the use of the 24 control switches when the computer is halted. The computer may be halted at any pre-determined state of the program counter through the use of the HALT position of control switches 9 through 23.

SEL 840 CONTROL PANEL



1. Raised to transfer bits 0 - 8 of the T register to the instruction register containing the operation code and address modifiers.
   Depressed to clear the instruction register.
2. Raised to transfer the contents of the T register to the selected register.
   Upper switch is depressed to clear the selected register.
   Lower switch depressed to clear T register.
3. Raised to connect switches 0 - 23 as console SENSE switches.
   Depressed to connect switches 9 - 23 as program HALT switches. (Computer halts when the program count equals selected value.)
4. Switches 0 - 23 are raised (lock) to function as SENSE or HALT switches and depressed to enter ONE bits into the T register.
5. Depressed to start computer operation.
6. Depressed to halt computer operation.
7. Depressed to clear all major registers and control latches.
8. Raised to repeat current instruction. Does not advance program counter.
   Depressed to execute single instruction in normal sequence and advances program counter.
9. Raised to enable SINGLE CYCLE switch to load contents of T register into memory.
   Depressed to enable SINGLE CYCLE switch to transfer contents of memory address to T register.
10. Depressed to release I/O wait and allow computer to resume.
11. Lights to indicate a wait for I/O function.
12. Raised to allow computer operation in the event of a memory parity error (lock).
    Centered to halt computer operation when memory parity error is detected.
    Depressed to reset the parity error latch.
13. Lights to indicate the detection of a memory parity error.
14. Depressed to inhibit operation of priority interrupts (lock).
15. Lights to indicate a priority interrupt.
16. Lights to indicate an arithmetic overflow condition.
17. Lights to indicate a program halt.
18. Pressed to apply power to computer.
    Lights when DC power is applied to computer.
19. Pressed to remove power from computer.
    Lights when AC power is connected to computer.
20. Indicators display the contents of the selected register.
21. Raised to gate the contents of the T register to the B accumulator and to display the contents of B accumulator register.
    Centered to display the contents of the T register.
    Depressed to gate the contents of the T register to Index Register 3 and to display the contents of index 3 register.
22. Raised to gate the contents of the T register and clear inputs to Index Register 1 and to display the contents of index 1 register.
    Centered to gate the contents of the T Register and clear inputs to the A accumulator and to display the contents of A accumulator register.
    Depressed to gate the contents of the T register and clear inputs to Index Register 2 and to display the contents of index 2 register.
23. Display the contents of the program counter.
24. Raised to enter bits 9 - 23 of the T register into the program counter.
    Depressed to clear the program counter.

Notes: 1. Only the SENSE, HALT and display selection switches are active while the computer is operating.
2. Switches 3 , 9 , 21 and 22 lock in all positions.

FUNCTIONAL DESCRIPTION

Memory Unit

The memory unit in the 840 computer is formed by one to eight magnetic core memory modules. Each module stores 2,048 or 4,096 or 8,192 24-bit computer words and is complete with its own drive electronics and registers. Each storage location is uniquely addressed through a memory address register and, when more than one module is used, each module has a unique address. Thus, when the full complement of up to eight 4K or four 8K modules is used, there are 32,768 separately addressed memory locations available to the computer.

Individual instruction and data words are loaded into specific addresses prior to the program execution. This may be done manually through the panel controls or automatically through the use of the supplied loader program. Each input word is transferred to the memory input register and the accompanying storage address is transferred to the memory address register. When both registers have been loaded, a "write" command is issued by the program control unit and the 24 bits in the memory data register are written into the 24 magnetic cores addressed by the memory address register.

When the entire group of instruction words forming a program is loaded and the computer started, addresses selected by the control unit are sent to the memory address register and a "read" command is issued. The status of each core at that address is sensed and transferred to the memory output register. The sensing of the cores sets them all to the same state, so the memory word now in the memory data register is immediately rewritten into its original memory location so as to be available for later use. The word is then transferred to the control unit to be decoded or to the arithmetic unit for computation. The memory read and write cycles are completely automatic so that only the memory address and source or destination must be supplied by the program.

Control Unit

The control unit contains a program counter which addresses the memory locations containing the instruction words. The program counter is initially set to the memory address containing the first instruction word and then automatically incremented to address successive words. The program counter may be set to an address out of the normal sequence by any one of several skip or branch instruction words provided for that purpose. Once set to a new address, the count is then sequentially incremented until again changed. The new address may be lower or higher than the last successive address so that the same program or portion of a program may be repeated many times. Such a repetitive program is called a "loop" which normally is repeated until some specific condition is detected by the control unit. When such a condition is found, the program counter is set to a different beginning address so that other portions of the program may be accomplished. It is these branch and skip instructions that allow the computer to

handle problems of great complexity requiring many separate instructions without using a vast number of memory locations merely to store the program.

The instruction words are unloaded from the memory addresses specified by the program counter. The memory cycle during which this occurs is referred to as the "Instruction Cycle". Some instruction words contain a memory address which specifies the location of an "operand" which is to be operated on by the computer. For these instructions, a second memory cycle, the "Execution Cycle", is required. During the execution cycle, the memory address is supplied by the "operand address" contained in the instruction word. The operand is read from memory and operated upon according to signals provided by the operation code.

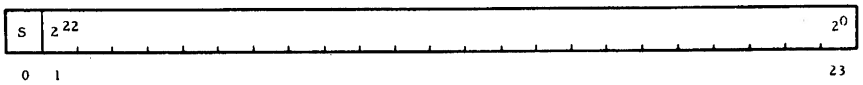
Many instruction words require no operand from memory and are executed completely within the instruction cycle. Others, while requiring no operand from memory, do require one or more execution cycles to complete. Chief among this latter group are the shift instructions which are used to rearrange the contents of data words. For these instructions, a group of bits within the instruction word defines the number of shifts to be performed while the operation code of the word defines the type of shifting to be done.

Arithmetic Unit

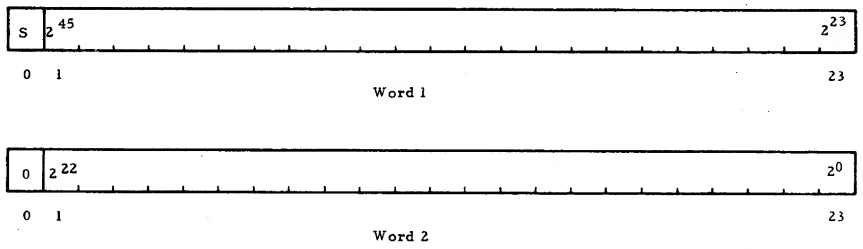
The arithmetic unit consists of a 24-bit adder and several accessory storage registers. Two of these registers, the A accumulator and the B accumulator, may be loaded and unloaded by instruction. The A accumulator is the primary arithmetic register and derives its name from its function of accumulating results of the arithmetic operations. Because only one word may be taken from the memory and input/output units by each instruction, the second operand in arithmetic operations must be loaded in a register prior to the selection of arithmetic add and subtract instructions. The A accumulator fulfills this function and also provides temporary storage for the result of the arithmetic operation. The B accumulator holds the multiplier during multiply operations and stores the least significant bits of the product. In addition to these strictly arithmetic functions, the two accumulators provide a convenient storage area for rearranging data words through shifting operations.

A third register connected to the adder is the "T register" which holds the operand unloaded from the memory. This 24-bit register and the 24-bit A accumulator both supply inputs to the 24-bit binary adder. The T register provides temporary storage for the addend, minuend, multiplicand and divisor during arithmetic operations.

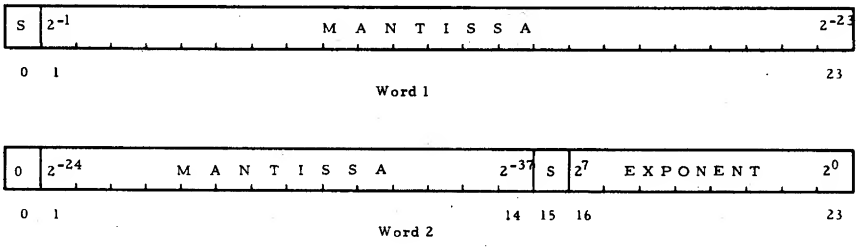
The basic data format of the SEL 840 computer is a 24-bit binary single-precision fixed point word (shown below). This format contains the sign bit in bit position 0 with bit position 1 holding the most significant data bit and bit position 23 holding the least-significant bit. This format is defined as an integer with an imaginary binary point located to the right of bit position 23. The 840 set of library integer subroutines recognizes this representation. The programmer may, of course, scale single-precision words in any desired manner and utilize the extensive shift and test instruction repertoire to maintain the binary point location.



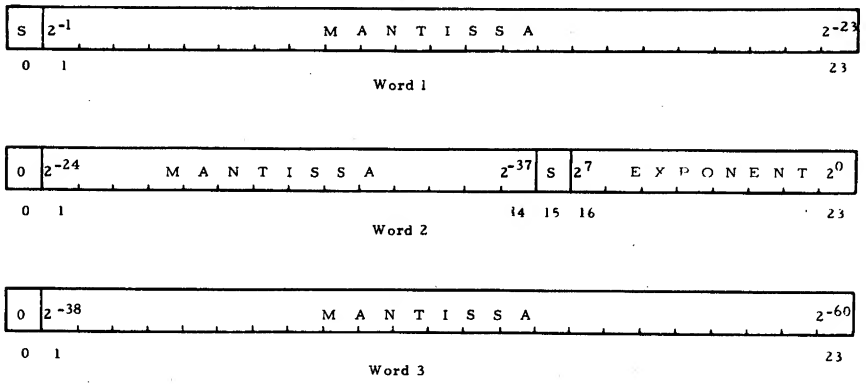
The SEL 840 also accommodates double-precision data words of 46 bits plus sign through the use of the extended (B) accumulator. Each double-precision data word must be stored in two adjacent memory locations with the most significant half stored in the first (lower) address. Each double-precision library subroutine utilizes the format shown below. The product generated by a single-precision mutiply is located in the A and B accumulators in this format. The dividend is assumed to be in this double-precision format prior to the execution of the DIVIDE instruction.



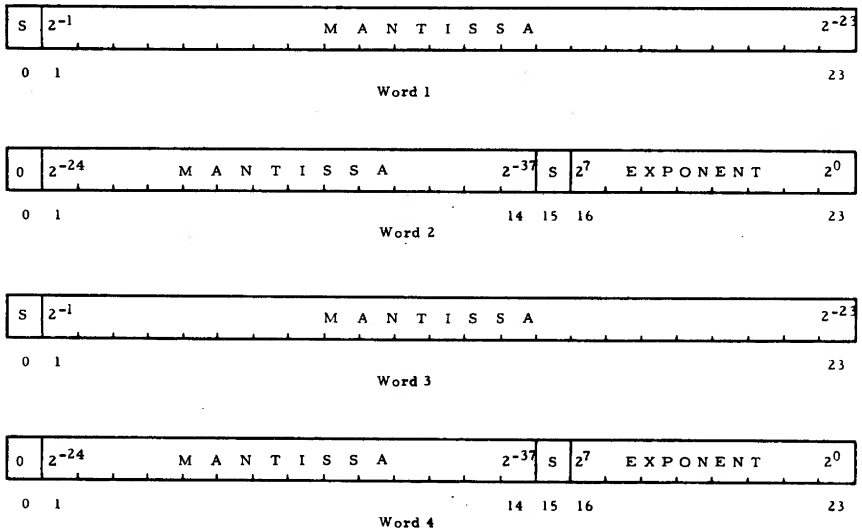
Three floating point data formats are utilized by the SEL 840 library. The single-precision floating point format (shown below) consists of two words. The first word contains the sign and 23 most significant bits of the fractional mantissa; the second word contains the 14 least significant mantissa bits and a signed 8-bit exponent. The words are stored in adjacent memory locations with the first word located in the lower memory address. Both the mantissa and the exponents carry separate signs so that the mantissa may be positive or negative independent of the sign of the exponent; thus, for example, the mantissa may be in the two's complement negative form while the exponent is in normal positive form.



A double-precision floating point format (shown below) consisting of three memory words is provided for use with the set of double-precision floating point library subroutines.



The third floating point data format (shown below) is provided for the set of FORTRAN IV subroutines dealing with complex numbers.



The arithmetic unit includes two control latches which are addressable by the program. The first of these is the OVERFLOW latch which can be set during addition, subtraction and division operations. The overflow for an add or subtract occurs when the result exceeds the accumulator capacity of ±8,388,607. A divide overflow occurs if the divisor is zero or if the divisor is equal to or smaller than the dividend. This latter overflow is due to the fact that the machine treats all divide arguments as double-precision numbers by scaling the single-precision divisor by 2^{23}. If the dividend is larger than the scaled divisor, the quotient will necessarily be a number greater than 2^{23}. Such a number exceeds the capacity of the 24 bit A accumulator in which the quotient is to be stored and thus produces a false divide.

The set overflow latch lights the OVERFLOW indicator on the control console and remains set until tested, and reset, by an BOF - Branch On Overflow-instruction. Because the latch remains set until tested, such a test should be made immediately following an arithmetic process when an overflow

condition could result. This prevents the possibility of a second overflow being undetected by the already set latch.

The second addressable arithmetic latch is the CARRY latch which connects to the least significant bit of the parallel adder. This latch is set in the regular arithmetic processes to produce a two's complement number (one's complement of the number plus one). The latch is used in the addition and subtraction of double-precision numbers formed in the A and B accumulators. The least significant words of the double-precision numbers are processed and stored in the B accumulator. If a carry or borrow is generated, it will cause the sign of the B accumulator to change. A CSB - Copy Sign of B - instruction is used to set the carry latch to the state of the B accumulator sign bit and then reset the B sign bit to ZERO (as required in the double-precision format). If the operation is addition, the true output of the carry latch is added together with the most significant words; if a subtraction operation is in process the false output of the carry latch is added to the most significant words (effectively subtracting the borrow). The carry latch is automatically reset after all instructions except the CSB instruction.

Input/Output Unit

The basic SEL 840 is equipped with two input/output channels - one for 24-bit word transfers and one for 6 or 8-bit character transfers. Both channels are connected to the computer buses which are used as the routes of transfer between computer units. Both basic I/O channels handle data under the direction of the control unit. In addition to the data transfer channels, the I/O unit also includes control and sense lines to and from the peripheral equipment.

Before communication can begin with any unit connected to either channel, the unit must be selected by a command from the control unit. The selected unit is then connected to the transfer bus and any previously connected unit is disconnected. The unit may then be commanded by the control unit to perform a function such as a read or write operation. The selection, connection and commanding of the I/O unit is controlled by a single input/output instruction. The status of the selected unit, active, inactive, stopped, etc., is determined through the use of test instructions.

The SEL 840 also contains a priority interrupt section which allows input/output units to interrupt the normal program. This arrangement is most useful to allow continuous operation of the program with interruptions occurring only when data is ready to be transferred. If the priority interrupt system were not used, the program would have to wait until a series of transfers were completed before resuming operation. With the priority interrupt, the computer may proceed with its normal program during the time which would otherwise be wasted.

An electric typewriter is supplied with the computer and connected to the character channel. The typewriter is used by the operator to communicate with the computer and, together with the control panel, allows manual direction of the computer. The typewriter also serves as a printer for the computer to communicate with the operator. In this use, previously prepared messages are transmitted from the computer when a program is completed, more data required, invalid data detected, etc. The operator may then supply data via the typewriter, prepare new inputs on other peripheral units or restart the program at a new point.

Extended Arithmetic Unit (EAU)

The optional extended arithmetic unit offers the means of rapidly and easily accomplishing the double precision and floating point arithmetic invariably associated with scientific computation. A distinct group of 22 instruction words are provided to program the EAU in single precision, double precision and floating point modes of add, subtract, multiply and divide operations. The results of EAU operations may be tested for sign, zero and overflow conditions. The floating point results may be obtained in either the normalized or unnormalized format. Any EAU operation initiated by the mainframe control unit proceeds independantly of the mainframe until that operation is completed; during this time no further such operations may be initiated and this "active" condition may be tested by a special instruction provided for that purpose. The EAU instruction repertoire includes several load/store instructions which transfer one word or two words between memory and the EAU depending on the programmed mode. In addition there are unique load/store instructions for formatting of single precision data used in EAU operations.

The extended arithmetic unit consists of two 48-bit accumulators (EA and EB), a 48-bit transfer register (ET) and a 48-bit parallel adder to accomodate double precision fixed point data and the mantissas of floating point data. The floating point exponents are processed in special 9-bit registers and a 9-bit parallel adder. The EAU arithmetic operations are performed using the same two's complement binary arithmetic used in the mainframe arithmetic unit.

Besides providing a fast, convenient method of accomplishing double precision and floating point computation, the EAU also allows operation on single precision data in parallel with the standard arithmetic unit. This capability finds its greatest value in high-speed real time operation of the SEL 840. In these applications, the standard arithmetic registers may be used to format input and output data words through the shift and scaling instruction provided for this unit while real time calculations are made in the EAU.

PROGRAMMING THE SEL 840 COMPUTER

The SEL 840 is operated by a series of instruction words stored in the magnetic core memory. The instruction words are successively read from memory locations addressed by the program counter. Each word specifies one operation - transferring a data word from an input channel to a memory location, adding a memory word to the word in the A accumulator, shifting the contents of the B accumulator, etc. The program counter is normally advanced one count after each instruction to access the instruction word located in the next sequential memory address. The program counter may be preset to any count by branch/skip instructions, which detect certain conditions such as A accumulator sign positive, overflow condition, input word ready, etc. The program counter then continues its sequential advance, but starting from the new address until again preset. The branch may be to either a higher or lower count so that portions of a program may be repeated until the branch condition is no longer present.

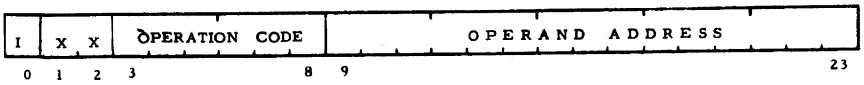
A list of 91 instructions is provided for the basic SEL 840 computer. This list includes Load/Store instructions which transfer words between the memory and the accumulators, Arithmetic instructions, Shift instructions which allow rearrangement of the bits within words, Logical instructions (AND, OR, NEGATE, etc.), Control instructions (HALT, etc.) Branch/Skip instructions to provide program modification and Input/Output instructions to command peripheral devices and transfer data into and out of the computer. Also included is a group of 22 instructions provided for use with the Extended Arithmetic Unit.

Each instruction word is formed by 24 bits, each of which performs a particular function - defining the operation to be performed, addressing a memory location, defining the number of shifts, etc. The function of a particular bit will vary in different types of instructions. For example, in some words, bit 14 forms part of a memory address; in others, bit 14 forms part of the operation code. The function of the bits depends on the instruction word type defined by the six -bit operation code located in bits 3 - 8 of the first word of each instruction.

There are three types of instruction words used by the SEL 840, those containing memory addresses within the instruction word, those containing additional code bits in lieu of the address bits and input/output instruction words which may consist of one, two, or three separate words forming one instruction.

Memory Reference Instructions

The memory reference instructions access the magnetic core memory for an operand. These words contain a six-bit binary operation code, a 15-bit memory address and three address modifiers as shown below.



Memory Reference Instruction Word

The INDEX flags (shown as XX in bit positions 1 and 2 of the word format diagram) are coded to add the contents of the hardware index register 1, 2 or 3 to the operand address (m). One of three index registers is selected by the states of bits 1 and 2 If the code is 00_2, no indexing is done; if 01_2, Index Register 1 is selected; if 10_2, Index Register 2 is selected; if 11_2, Index Register 3 is used. These registers may be loaded with any positive or negative 15-bit number ranging from 00000_8 to 77777_8. The addition of this number to the operand address allows the addressing of any memory location within the full-size 32,768 - address memory. If, for example, the operand address is 04131_8 and the selected index count is 02022_8, the resulting effective address is 04131_8 (m) + 02022_8 (X) or 06153_8.

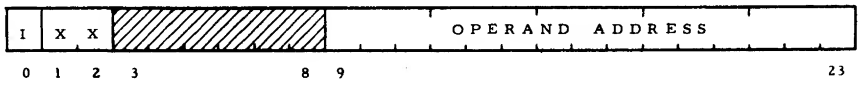
The Index registers serve another important function in that the registers may be incremented (increased by one) by instruction. The incrementing instruction also tests the register for a negative sign and generates a skip (an extra advance count to the program counter) if the register is not negative. This feature allows the programmer to load a negative number into an index register, add the proper Index flag to the basic instruction and create a minor iterative sub-program that will access a series of sequential memory addresses. Such a sub-program or "loop", written in assembly language, is shown below. (In assembly language, a ",1" is used to indicate an indexed instruction using index register 1, an "," indicates an octal number and either absolute (15432) or symbolic (LOOP, IPUT) addresses may be used. A complete description of the assembly language is presented in the SEL 840 assembler reference manual.)

Location	Operation	Address	Comments
INDX	LIX	='-20,1	Load an index count of -20_8 into Index Register 1.
LOOP	MIC	A;W	Transfers a series of input words from I/O channel A to a group of 20_8 memory locations ending with address 15431 (1 less than the specified address) with a wait flag.
	DAC	15432,1	
	IIB	LOOP,1	Increment (index) and branch to loop if the index count is not negative.
PROG	LAA	IPUT	Next instruction of regular program.

This series of instructions causes the first input word to be loaded into address 15412_8 (15432_8-20_8). The negative index count is then incremented by one ($-20_8 + 1 = -17_8$) by the IIB instruction which also tests the index register for a negative sign. The sign is negative for the first $16(20_8)$ increments so the branch back to the LOOP instruction is executed 16 times. The

execution of the MIC instruction adds the index count to the operand address of 15432_8-(X) to load the remaining words into addresses 15413_8-15431_8. The loop is repeated 16 (20_8) times to load the input words into addresses 15412_8 through 15431_8 before the index count of (+00) is produced by the IIB instruction. The IIB then advances the program counter to the next instruction of the normal program instead of branching to the LOOP instruction because of the positive sign of the index register. Note that the operand address must be one greater than the last address to be loaded because the skip occurs before (+00) is added to the operand address. Indexing requires no additional program time.

The INDIRECT flag (bit 0) does not alter the operand address as such, but instead changes the function of the contents of that address. Except when the indirect flag is used, the contents of the memory location accessed by the effective (m+X) address form the operand used in the operation specified by the instruction. The Indirect flag, however, uses the contents of the memory word to address the operand. In order to accomplish this function, a special word format is provided for indirect addresses.

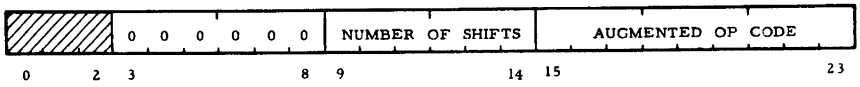


Indirect Address Word

The indirect word format contains 15 address bits and the Index flags which, if ONE's, add the index count to the indirect address. The index count may be added to the address in the instruction word, the indirect address or both or neither depending on the presence or absence of Index bits in the instruction and indirect words. The indirect address also includes an Indirect flag bit permitting multi-level indirect addressing. Each Indirect flag requires an additional 1.75 microsecond cycle to call the next word from memory.

Augmented Instructions

Many of the instruction words in the SEL 840 computer are augmented instructions with a 00_8 operation code. These instructions contain no memory address bits, but do contain up to nine additional (augmenting) operation code bits. The augmented 008 instruction word format is shown below.



Augmented 00_8 Instruction Word

The detection of the 00_8 operation code in the instruction register gates the nine augment code bits into a special decoding matrix. When these bits specify a shift operation, the shift count bits, bits 9 through 14, are loaded into a binary down-counter. The presence of bits in this counter gates shift pulses to the A and B accumulators and to the down-counter. When the count reaches zero, the shifting operation is complete.

Many of the branch/skip instructions are memory reference instructions that designate the memory location to which the program counter will be set if a certain condition is found to exist. The remainder of the branch/skip group are skip instructions which require no address bits. Each of the skip instructions tests for specific conditions within the computer - primarily within the arithmetic unit. If a condition is found to exist, the next sequential instruction is skipped, if the condition is absent, the next sequential instruction is executed. The next sequential instruction is usually an unconditional branch to some program loop which affects the condition tested by the skip instruction. Thus, the skip instruction, in conjunction with an unconditional branch instruction, serve as conditional branches. The skip/branch group includes two three-way skips, one of which is an augmented 00_8 instruction that skips none, one, or two sequential instructions depending on the contents of the A accumulator. The second is a memory address instruction which compares the value of a specified memory word and the A accumulator and then skips none, one or two sequential instructions depending on the result of the comparison.

Four other augmented operation codes, the 13_8, 17_8, 21_8 and 43_8 codes, are also augmented with additional code bits. These instructions, used primarily for input/output operations, have word formats that vary slightly one from another and may include two or three words to complete the instruction.

The augmented 13_8 operation code words are sense instructions which are used to test external conditions of I/O channels and units in a manner similar to the action of the internal skip instructions. The condition, the channel, and the unit to be tested are all defined by the augmenting code bits contained in the instruction word. If the tested-for condition is present, the next sequential instruction, usually an unconditional branch instruction, is skipped. The programming of the branch instruction allows the test instruction to be repeated until the required condition is present before continuing with the rest of the program.

The augmented 17_8 instructions contain additional code bits defining the function of the instruction, I/O unit control codes, words/block and characters/word counts and a wait flag. The wait flag allows the programmer the option of waiting for an input/output operation to be performed before continuing with the program, if that particular operation is vital, or to continue without waiting, if the remainder of the program is not dependent on the operation. There are two sets of word transfer instructions that allow transfers of words to and from the accumulator or memory and any specified channel. If the wait flag is not programmed for these instructions, the next sequential instruction will be skipped if

the channel is ready to receive or transmit data. The next instruction, executed if the channel is not ready, is usually an unconditional branch to allow the computer to perform other operations while the I/O device is busy. This construction allows the programmer the option of either stopping the program until an I/O device is ready to transmit or receive data or of continuing the program and periodically testing the I/O device, transferring a word after each test which indicates the device is ready.

Section II SEL 840 MACHINE LANGUAGE INSTRUCTIONS

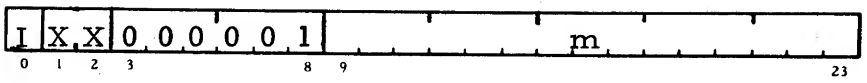
Each of the instruction words used in the SEL 840 computer is described in detail on the following pages. The descriptions are arranged in Load/Store, Arithmetic, Branch/Skip, Logical, Register Change, Shift, Control, Input/Output and EAU groups. Each instruction is identified in bold print by its mnemonic and operation code - whole numbers for memory address instruction (01, 16, etc.) hyphenated three-digit numbers for augmented 00_8 instructions (-000, -011, etc.) and whole numbers plus a decimal for the other augmented codes (17.31, 21.11, 43.1, etc.). The bit assignments for each instruction word are shown and, in the case of the two-and three-word instructions, described. The unassigned bit locations, shown as diagonal lines in the word format illustrations, should be set to zero.

Load/Store Instructions

Five of these instructions provide the means of accessing memory both to call data to the arithmetic registers for computation and to return the computed results. Two other instructions are provided to transfer data between any of the three index registers and the memory. These six instructions are memory reference words, while the last instruction in the group, the LCS instruction, is used to transfer the contents of the 24 console switches into the A accumulator. This instruction allows the switches to be used for manual modification of the program at pre-determined points.

LAA Load A Accumulator 01

Word Format:



Description: The contents of the effective memory address replace the previous contents of the A accumulator. The contents of memory are unchanged.

Note: The A accumulator must be loaded with the augend, subtrahend and most significant bits of the dividend prior to add, subtract and divide instructions.

Address Modifiers: Indirect and Index (1, 2 or 3)

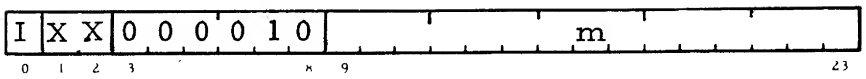
Timing: 2 Cycles

Registers Affected: A accumulator

Indicators: None

LBA Load B Accumulator 02

Word Format:



Description: The contents of the effective memory address replace the previous contents of the B accumulator. The contents of the memory are unchanged.

Note: The B accumulator must be loaded with the multiplier prior to a multiply instruction and with the least significant bits of the dividend prior to divide instructions.

Address Modifiers: Indirect and Index (1, 2 or 3)

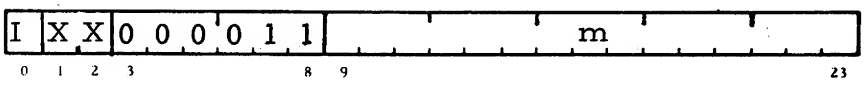
Timing: 2 Cycles

Registers Affected: B accumulator

Indicators: None

STA Store A Accumulator 03

Word Format:



Description: The contents of the A accumulator replace the previous contents of the effective memory address. The contents of the A accumulator are unchanged.

Address Modifiers: Indirect and Index (1, 2 or 3)

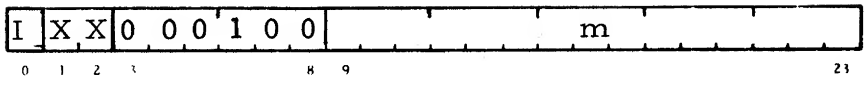
Timing: 2 Cycles

Registers Affected: None

Indicators: None

STB Store B Accumulator 04

Word Format:



Description: The contents of the B accumulator replace the previous contents of the effective memory address. The contents of the B accumulator are unchanged.

Address Modifiers: Indirect and Index (1, 2 or 3)

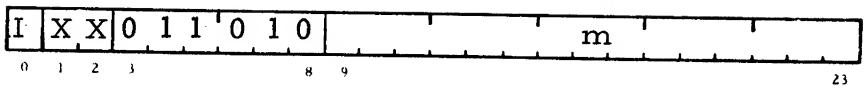
Timing: 2 Cycles

Registers Affected: None

Indicators: None

LIX — Load Index Register — 32

Word Format:



Description: The contents of the effective memory address replace the previous contents of the designated index register. The contents of memory are unchanged.

Note: The indirect address modifiers (bits 1 and 2) function as modifiers only if an indirect flag (bit 0) is present in this instruction; otherwise, bits 1 and 2 designate the index register that is to be the destination register. If indirect chaining is employed, the last indirect address (the one NOT containing an indirect flag) must specify the register to be loaded, while all other indirect addresses in the chain may specify any index register as a modifier.

Address Modifiers: Indirect

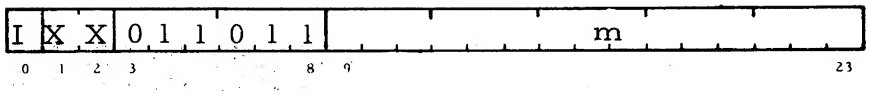
Timing: 2 Cycles

Registers Affected: 1, 2 or 3 Index register

Indicators: None

STI Store Index 33

Word Format:



Description: The contents of the designated index register replace the previous contents of the effective memory address. The contents of the index registers are unchanged.

Note: The indirect address modifiers (bits 1 and 2) function as modifiers only if an indirect flag (bit 0) is present in this instruction; otherwise, bits 1 and 2 designate the index register that is to be the source register. If indirect chaining is employed, the last indirect address (the one NOT containing an indirect flag) must specify the register to be stored, while all other indirect addresses in the chain may specify any index register as a modifier.

Address Modifiers: Indirect

Timing: 2 Cycles

Registers Affected: None

Indicators: None

LCS Load Control Switches 57

Word Format:

Description: The states of switches 0 through 23 on the control console replace the previous contents of the A accumulator (up="1"; center = "0").

Address Modifiers: None

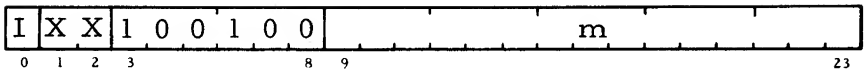
Timing: 1 Cycle

Registers Affected: A accumulator

Indicators: None

IAM — Interchange A Accumulator and Memory — 44

Word Format:



Description: The contents of the effective memory address replace the contents of the A accumulator and the contents of the A accumulator replace the contents of the effective memory address.

Address Modifiers: Indirect and Index (1, 2 or 3)

Timing: 3 Cycles

Registers Affected: A accumulator

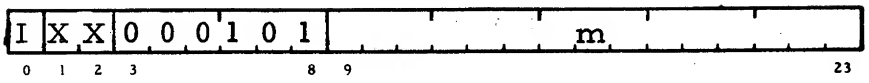
Indicators: None

Arithmetic Instructions

The arithmetic group of instructions include two add instructions, one, the AMA instruction, stores the sum in the A accumulator and a second, the AAM instruction, stores the sum in the memory location originally holding one of the operands. The SMA instruction subtracts the contents of a memory location from the contents of the A accumulator. The MPY instruction multiplies the single precision contents of a memory location by the single precision contents of the B accumulator to produce a double precision product in the A and B accumulators. The DIV instruction divides the double precision dividend in the A and B accumulators by the single precision contents of a memory location and stores the single precision quotient in the A accumulator. The RNA instruction is used to round the contents of the A accumulator by the magnitude of the contents of the B accumulator. A third add instruction, the AMX instruction, adds the 15 least significant bits of the addressed memory word to the index register selected by the instruction.

AMA — Add Memory to A Accumulator — 05

Word Format:



Description: The contents of the effective memory address (the addend) are algebraically added to the contents of the A accumulator (the augend). The sum is stored in the A accumulator and the sign of the A accumulator is set to the algebraic sign of the sum. The contents of the memory are unchanged.

Note: The A accumulator must be loaded with the augend prior to the AMA instruction, either by a LAA instruction or as the result of a previous operation involving the A accumulator.

Address Modifiers: Indirect and Index (1, 2 or 3)

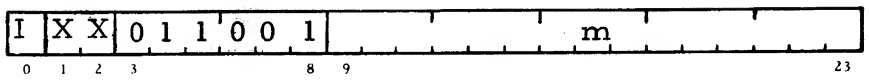
Timing: 2 Cycles

Registers Affected: A accumulator

Indicators: OVERFLOW if the sum exceeds 23 bits plus sign.

AAM Add A Accumulator to Memory 31

Word Format:



Description: The contents of the A accumulator (addend) are algebraically added to the contents of the effective memory address (augend). The sum replaces the augend in the effective memory address. The contents of the A accumulator are unchanged.

Note: This instruction is useful in the adding of a constant, stated or derived, to a series of related data words.

Address Modifiers: Indirect and Index (1, 2 or 3)

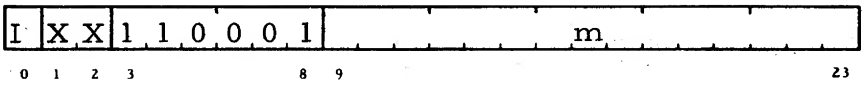
Timing: 3 Cycles

Registers Affected: None

Indicators: OVERFLOW if the sum exceeds 23 bits plus sign.

AMX Add Memory to Index Register 61

Word Format:



Description: The contents of the effective memory address (the addend) are algebraically added to the contents of the selected index register (the augend). The 15 least significant bits of the sum are stored in the selected index register. The contents of the memory are unchanged.

Note: The indirect address modifiers (bits 1 and 2) function as modifiers only if an indirect flag (bit 0) is present in this instruction; otherwise, bits 1 and 2 designate the index register that is to be the destination register. If indirect chaining is employed, the last indirect address (the one NOT containing an indirect flag) must specify the register to be loaded; while all other indirect addresses in the chain may specify any index register as a modifier.

Address Modifiers: Indirect

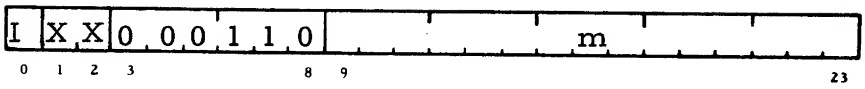
Timing: 2 Cycles

Registers Affected: Index register 1, 2 or 3

Indicators: None

SMA Subtract Memory from A Accumulator 06

Word Format:



Description: The contents of the effective memory address (the subtrahend) are algebraically subtracted from the contents of the A accumulator (the minuend). The difference replaces the minuend in the A accumulator and the sign of the A accumulator is set to the sign of the algebraic sum. The contents of memory are unchanged.

Note: The A accumulator must be loaded prior to the execution of the SMA instruction, either through an LAA instruction or as the result of a previous operation involving the A accumulator.

Address Modifiers: Indirect and Index (1, 2 or 3)

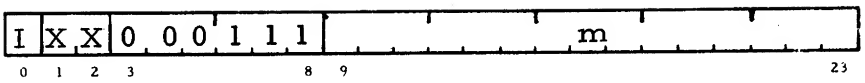
Timing: 2 Cycles

Registers Affected: A accumulator

Indicators: OVERFLOW if the difference exceeds 23 bits plus sign.

MPY Multiply 07

Word Format:



Description: The contents of the effective memory address (multiplicand) are multiplied by the contents of the B accumulator (multiplier). The most significant half of the product replaces the previous contents of the A accumulator. The least significant half of the product replaces the previous contents of the B accumulator. The sign of the A accumulator is determined by the algebraic sign of the product; the sign of the B accumulator is set to plus. The contents of the memory are unchanged.

Note: If the multiplier and multiplicand are considered to be integers (binary point to the right of bit 23), the product is a double precision integer (binary point to the right of bit 46 in the B accumulator). If the multiplier, multiplicand, or both are scaled left 2^m and 2^n, then the product is scaled left 2^{m+n}.

Address Modifiers: Indirect and Index (1, 2 or 3)

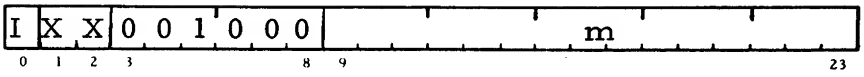
Timing: 14 Cycles

Registers Affected: A accumulator, B accumulator

Indicators: None

DIV Divide 10

Word Format:



Description: The contents of the A and B accumulators (double length dividend) are divided by the contents of the effective memory address (single length divisor). The quotient is stored in the A accumulator and the remainder is stored in the B accumulator. The sign of the quotient is set to the algebraic sum of the divisor and dividend signs. The sign of the remainder is set to the sign of the original dividend. The contents of the memory are unchanged.

Note: The double length dividend must be loaded into the A and B accumulators in standard double precision format prior to the call of the divide instruction. If a single precision dividend is to be used, it is normally loaded in the A accumulator. The contents of the B accumulator must be set to zero prior to call of the divide or the answer will be in error. Since the 840 considers all dividends to be double length, loading a single precision number into the A accumulator will, by definition, scale the number by 2^{23}. If the divisor is smaller than the single precision dividend a divide overflow will occur because the quotient will be larger than 2^{23}. If the programmer desires to scale the single precision dividend down by placing it in the B accumulator, the A accumulator must be cleared prior to the call of the divide. If the single precision number is an integer and is scaled at 2^0 (placed in B accumulator) all fractional parts of the true quotient will be discarded. If the double precision dividend is scaled left 2^m and the single precision divisor is scaled left 2^n, the quotient is scaled left 2^{m-n}. To avoid a divide overflow, the divisor and dividend must be scaled so that the quotient $N \times 2^{m-n}$ does not exceed $2^{23}-1$.

Address Modifiers: Indirect and Index (1, 2 or 3)

Timing: 15 Cycles

Registers Affected: A accumulator, B accumulator

Indicators: OVERFLOW if divisor exceeds dividend or if divisor is equal to zero.

RNA Round A Accumulator **60**

Word Format:

Description: The contents of the A accumulator are increased by one if B_1 in the B accumulator is an ONE and the sign of the A accumulator is positive. The contents of the A accumulator are decreased by one if bit B_1 in the B accumulator is a ZERO and the sign of the A accumulator is negative.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: A accumulator

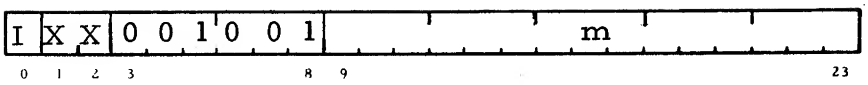
Indicators: None

Branch/Skip Instructions

This group of instructions provide the decision-making capability of the computer. The majority of these instructions are branch instructions which pre-set the program counter to the effective address contained in the instruction, while the remainder are skip instructions which advance the program counter one or two additional counts. Three of the branch instructions, the BRU, SPB and PIR instructions, are unconditional branches, but the BAZ, BAP, BAN, IIB and BOF instruction branch only if certain conditions exist. These conditions include negative or positive signs of the A accumulator or memory, zero contents in the A accumulator, an arithmetic overflow or a negative index count. The SMP, CMA, SNS and SAS skip instructions are all conditional instructions testing the sign of the contents of a specific memory word or the A accumulator, testing the status of a console sense switch or comparing the contents of the A accumulator and a specific memory word. The SAS - skip on A sign - and CMA - compare Memory and A - are both three-way skip instructions which advance the program counter one, two or three counts depending on the result.

BRU Unconditional Branch 11

Word Format:



Description: The effective memory address replaces the previous contents of the program counter. This causes the program to begin with the instruction located at the new memory address and proceed through sequential addresses from that point.

Address Modifiers: Indirect and Index (1, 2 or 3)

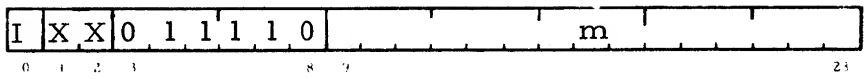
Timing: 1 Cycle

Registers Affected: Program counter

Indicators: None

PIR Priority Interrupt Return **36**

Word Format:



Description: The contents of the effective memory address replace the previous contents of the program counter and reset the highest set priority interrupt latch.

Address Modifiers: Indirect and Index (1, 2 or 3)

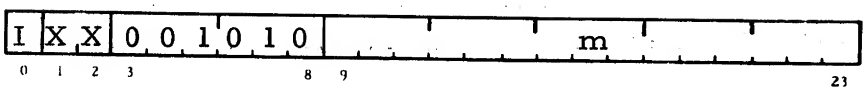
Timing: 2 Cycles

Registers Affected: Program counter

Indicators: None

SPB Store Place and Branch 12

Word Format:



Description: The contents of the program counter replace the previous contents of the effective memory address and the effective memory address plus 1 replaces the previous contents of the program counter.

Note: The SPB instruction is ordinarily used to enter a subroutine while providing the means of returning to the main program. The first word of the subroutine (effective address) holds the return address (program count) while the second word of the subroutine (effective address + 1) holds the first subroutine instruction. The last instruction of the subroutine is a BRU Indirect to the first word to replace the stored program count in the program counter.

Address Modifiers: Indirect and Index (1, 2 or 3)

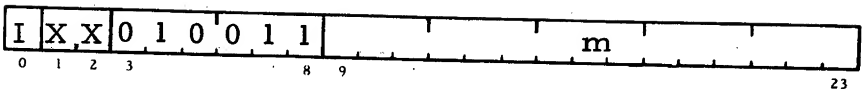
Timing: 2 Cycles

Registers Affected: Program counter

Indicators: None

BAN Branch on Negative 23

Word Format:



Description: If the contents of the A accumulator are less than zero, the effective address replaces the previous contents of the program counter. If the contents of the A accumulator are greater than, or equal to, zero, the next sequential instruction is executed.

Address Modifiers: Indirect and Index (1, 2 or 3)

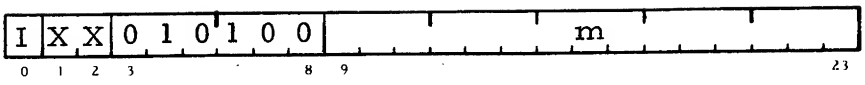
Timing: 1 Cycle

Registers Affected: Program counter

Indicators: None

BAP Branch on Positive 24

Word Format:



Description: If the contents of the A accumulator are greater than, or equal to, zero, the effective address replaces the previous contents of the program counter. If the contents of the A accumulator are less than zero, the next sequential instruction is executed.

Address Modifiers: Indirect and Index (1, 2 or 3)

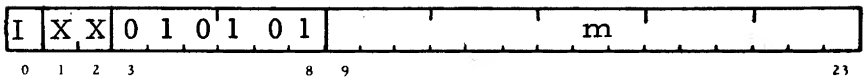
Timing: 1 Cycle

Registers Affected: Program counter

Indicators: None

BOF Branch on Overflow 25

Word Format:



Description: If the OVERFLOW latch is set, the effective address replaces the previous contents of the program counter and resets the OVERFLOW latch.

If the OVERFLOW latch is not set, the next sequential instruction is executed.

Address Modifiers: Indirect and Index (1, 2 or 3)

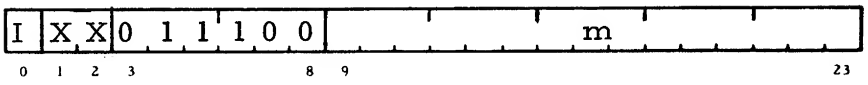
Timing: 1 Cycle

Registers Affected: Program counter

Indicators: None

IIB Increment Index and Branch 34

Word Format:



Description: The contents of the designated index register are increased by one. If the contents of that register are then non-zero the effective address replaces the previous contents of the program counter. If the contents of that register are zero negative, the next sequential instruction is executed.

Note: Bits 1 and 2, that normally specify which index register will be used to modify the address, instead designate the index register that will be incremented.

Address Modifiers: Indirect

Timing: 1 Cycle

Registers Affected: Program counter, Index register (1, 2 or 3)

Indicators: None

SAS Skip on Accumulator Sign **-021**

Word Format:

Description: The contents of the accumulator are tested. If the contents are negative, the next sequential instruction (NI) is executed. If the contents are zero, the second sequential instruction (NI+1) is executed. If the contents are positive, the third sequential instruction (NI+2) is executed.

Address Modifiers: None

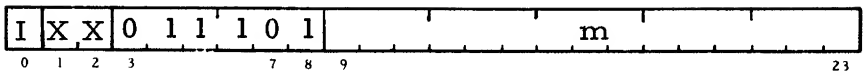
Timing: 1 Cycle

Registers Affected: Program counter

Indicators: None

SMP Skip if Memory Positive 35

Word Format:



Description: If the contents of the effective memory address are greater than, or equal to, zero, the next sequential instruction (NI) is skipped and the second sequential instruction (NI+1) is executed. If the contents of the memory address are less than zero, the next sequential instruction (NI) is executed.

Address Modifiers: Indirect and Index (1, 2 or 3)

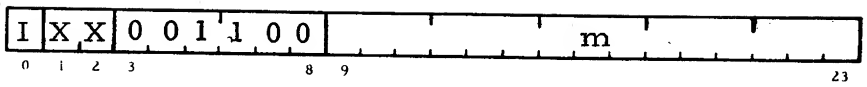
Timing: 2 Cycles

Registers Affected: Program counter

Indicators: None

IMS — Increment Memory and Skip — 14

Word Format:



Description: The contents of the effective memory address are increased by one. If the contents then equal zero, the next instruction (NI) is skipped and the second sequential instruction (NI+1) is executed. If the contents of the effective address do not equal zero, the next instruction (NI) is executed.

Note: This instruction allows any memory location to be used as an auxiliary index register if that address is loaded with the negative index count.

Address Modifiers: Indirect and Index (1, 2 or 3)

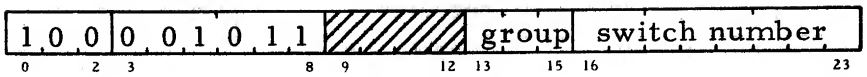
Timing: 3 Cycles

Registers Affected: Program counter

Indicators: None

SNS Skip No Signal 13.4

Word Format:



Description: If the designated console control switch is set, the next instruction (NI) is executed; if that switch is not set or if the SENSE/HALT switch is not in the SENSE position, the next instruction (NI) is skipped and the second sequential instruction (NI+1) is executed.

Instruction word bits 13 through 23 designate which of the 24 control switches 0-23 is to be tested. Bit 13 designates switches 0-7; bit 14 designates switches 8-15; bit 15 designates switches 16-23. Bits 16 through 23 of the instruction word then designate which of the eight switches in the selected group are to be tested. For example: bits 14 and 23 are set to ONE and bits 13 and 15 through 22 are set to ZERO to designate switch 15 as the SENSE switch to be tested. If more than one switch is selected in the instruction, then the skip will occur if any selected switch is set.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: Program counter

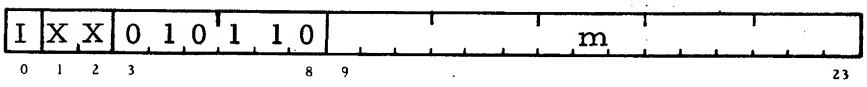
Indicators: None

Logical Instructions

The six logical instructions affect only the A and B accumulators. These instructions are provided to allow the logical modification of instruction and data words. The MAA, MOA and MEA instructions are used to mask (logically remove) portions of a single word, merge (logically combine) and compare (logically exclusive OR) two words. The contents of the A accumulator may be two's complemented through the NEG instruction, while the sign bit of the A accumulator can be complemented by the ASC instruction. Sign magnitude form numbers (true binary form with either + or - sign to show polarity) can be converted to two's complement form (one's complement + 1 and - sign for negative numbers) and the reverse by the CNS instruction. The NEG, ASC and CNS are used primarily in the arranging of data formats to satisfy input/output requirements, but also find use in creating special flag, indicator and constant words.

MEA Memory EXCLUSIVE OR A Accumulator 26

Word Format:



Description: The contents of the effective memory address and the contents of the A accumulator form bit-by-bit (no carry) arithmetic sum which is stored in the A accumulator.

EXCLUSIVE OR Truth Table		
(M)	(A)	A
0	0	0
1	0	1
0	1	1
1	1	0

Note: This instruction may be used as a comparison operation to detect specific locations of like and unlike bits.

The bits in the same bit locations must be different to produce a "1" in the result.

For example:

101	101	101	010	101	010	101	010	A acc. word
101	010	101	110	011	010	101	010	Mem. word
000	111	000	100	110	000	000	000	Log. Result

Address Modifiers: Indirect and Index (1, 2 or 3)

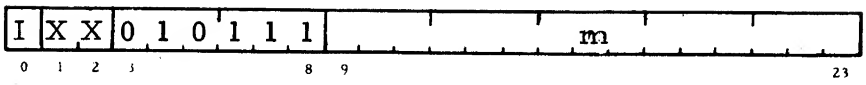
Timing: 2 Cycles

Registers Affected: A accumulator

Indicators: None

MAA Memory AND A Accumulator 27

Word Format:



Description: The contents of the effective memory address and the contents of the A accumulator form a logical product which is stored in the A accumulator.

Logical AND Truth Table		
(A)	(M)	A
0	0	0
1	0	0
0	1	0
1	1	1

Note: This instruction may be used in masking operations to separate or delete portions of a computer word.

A "1" must occur in the same bit position of each word for a "1" to result.

For example:

000	000	000	000	111	111	111	111	A acc. word
101	010	101	010	101	010	101	010	Mem. word
000	000	000	000	101	010	101	010	Log. Product

Address Modifiers: Indirect and Index (1, 2 or 3)

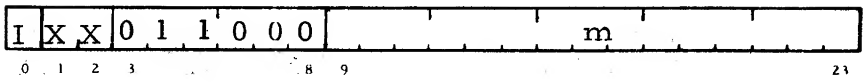
Timing: 2 Cycles

Registers Affected: A accumulator

Indicators: None

MOA Memory OR A Accumulator 30

Word Format:



Description: The contents of the effective memory address and the contents of the A accumulator form a logical sum which is stored in the A accumulator.

Logical OR Truth Table		
(A)	(M)	A
0	0	0
1	0	1
0	1	1
1	1	1

Note: This instruction may be used in merging operations to combine separate portions of two computer words.

A "1" in a given bit position of either word produces a "1" in the result.

For example:

000	000	000	010	111	111	111	111	A acc. word
101	010	101	010	101	010	101	010	Memory word
101	010	101	010	111	111	111	111	Logical Sum

Address Modifiers: Indirect and Index (1, 2 or 3)

Timing: 2 Cycles

Registers Affected: A accumulator

Indicators: None

ASC Complement A Sign -020

Word Format:

Description: The A accumulator sign bit (A_0) is complemented.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: A accumulator

Indicators: None

NEG Negate A Accumulator 56

Word Format:

Description: The contents of the A accumulator are two's complemented.

Address Modifiers: None

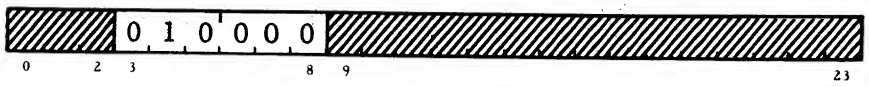
Timing: 1 Cycle

Registers Affected: A accumulator

Indicators: None

CNS Convert Number System 20

Word Format:



Description: Bits 1 through 23 of words in the A accumulator are two's complemented. The sign is unchanged. If the word was originally in the sign magnitude form, it is converted to two's complement form. If originally in two's complement form, it is converted to sign magnitude form. Positive-signed words are unchanged since the word structure is the same for both forms.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: A accumulator

Indicators: None

Register Change Instructions

These six instructions are used primarily to manipulate data, create specific formats and to perform routine operations connected with double precision arithmetic.

The TBI instruction transfers the contents of the B accumulator to the selected index register. The TBA and TAB instructions transfer the contents of the B accumulator to the A accumulator and the reverse, while leaving the contents of the source register intact. The IAB instruction interchanges the contents of the A and B accumulators changing both. The CLA instruction sets the contents of the A accumulator to zero. The CSB instruction copies the sign of the B accumulator into the CARRY latch and sets the B accumulator sign to zero.

TBI Transfer B Accumulator to Index Register **-002**

Word Format:

Description: The contents of the designated Index register are replaced by the contents of the B accumulator. The contents of the B accumulator are unchanged.

Note: Bits 1 and 2 that normally designate which Index register will be used to modify the address, instead designate which Index register will be the destination of the transfer.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: Index register (1, 2 or 3)

Indicators: None

CLA Clear A Accumulator -003

Word Format:

Description: The contents of the A accumulator are destroyed and replaced with all ZERO's.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: A accumulator

Indicators: None

TBA Transfer B Accumulator to A Accumulator **-004**

Word Format:

Description: The contents of the B accumulator replace the contents of the A accumulator. The contents of the B accumulator are unchanged.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: A accumulator

Indicators: None

TAB — Transfer A Accumulator to B Accumulator — -005

Word Format:

Description: The contents of the A accumulator replace the previous contents of the B accumulator. The contents of the A accumulator are unchanged.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: B accumulator

Indicators: None

IAB — Interchange A and B Accumulators — -006

Word Format:

Description: The contents of the A accumulator are replaced by the contents of the B accumulator and the contents of the B accumulator are replaced by the contents of the A accumulator.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: A accumulator, B accumulator

Indicators: None

CSB Carry Sign of B Accumulator -007

Word Format:

Description: The sign of the B accumulator is transferred to the CARRY latch and replaced with a ZERO (plus).

Note: This instruction is used with double precision arithmetic to obtain the proper double precision word format.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: B accumulator

Indicators: None

Shift Instructions

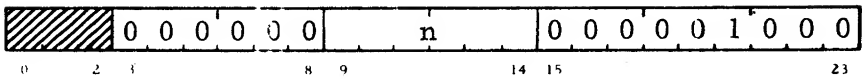
The nine instructions forming the shift group are augmented 00_8 instructions with bits 9 through 14 containing the binary shift count. While the actual count is in binary code, the number of shifts is usually specified in decimal in symbolic coding. Up to 63 (77_8) shifts may be programmed but only 48 (60_8) are required to fully rotate both the A and B accumulators.

There are two types of shift instructions; arithmetic shifts which bypass the sign bit, and logical shifts which move all 24 bits. Right arithmetic shifts move bits from position 1 to 2, 2 to 3, 3 or 4, etc. with bit 1 set to the state of the sign bit and the bit originally located in bit 23 is shifted off. In left arithmetic shifts the bits are moved from positions 23 to 22, 22 to 21, 21 to 20, etc. with ZERO's loaded into bit 23 and the original bits shifted off bit position 1. The sign bit remains intact. In right logical shifts, the sign bit is shifted to position 1, 1 to 2, 2 to 3, etc. and ZERO's are shifted into the sign position. In left logical shifts, ZERO's are loaded into bit position 23, 23 to 22, 22 to 21, etc. and the sign bit is shifted off.

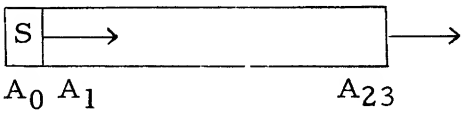
Both accumulators may be shifted together in right arithmetic, left logical, left logical rotate and left normalize modes. The rotate instruction moves the sign bit of the A accumulator to bit position 23 of the B accumulator as the other bits are moved left from B to A. The normalize instruction left shifts bits 1 through 23 of the B accumulator and bits 0 through 23 of the A accumulator until the bits in the A accumulator sign position and position 1 are unlike.

RSA Right Shift A Accumulator -010

Word Format:



Description: Bits A_1 through A_{23} of the A accumulator are shifted right n places as specified by bits 9 through 14 of the instruction word. The sign bit is unchanged, but does supply bits (ONE's if negative, ZERO's if positive) to bit position A_1 as the most significant bits are shifted right. The least significant bits are shifted off bit position A_{23}.



Address Modifiers: None

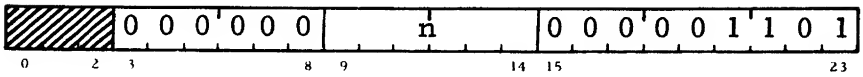
Timing: The time required to complete this instruction varies with the number of programmed shifts as follows:

1	- 4	Shifts	- 2 Cycles
5	- 8	Shifts	- 3 Cycles
9	- 12	Shifts	- 4 Cycles
13	- 16	Shifts	- 5 Cycles
17	- 20	Shifts	- 6 Cycles
21	- 23	Shifts	- 7 Cycles

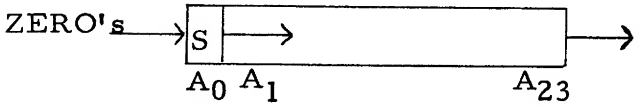
Registers Affected: A accumulator

Indicators: None

Word Format:



Description: Bits A_0 through A_{23} of the A accumulator are shifted right n places as specified by bits 9 through 14 of the instruction word. ZERO's are shifted into bit position A_0 to replace the most significant bits as the least significant bits are shifted off bit position A_{23}.



Address Modifiers: None

Timing: The time required to complete this instruction varies with the number of programmed shifts as follows:

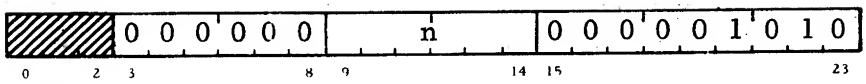
1	- 4	Shifts	- 2 Cycles
5	- 8	Shifts	- 3 Cycles
9	- 12	Shifts	- 4 Cycles
13	- 16	Shifts	- 5 Cycles
17	- 20	Shifts	- 6 Cycles
21	- 23	Shifts	- 7 Cycles

Registers Affected: A accumulator

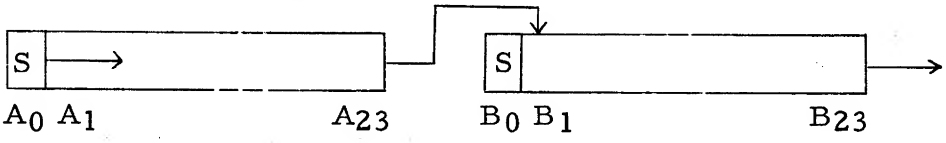
Indicators: None

FRA Full Right Arithmetic Shift -012

Word Format:



Description: Bits A_1 through A_{23} and B_1 through B_{23} of the A and B accumulators are shifted right n places as specified by bits 9 through 14 of the instruction word. Neither sign bit is shifted, but the A accumulator sign supplies bits (ONE's if negative, ZERO's if positive) to bit position A_1 as the most significant A accumulator bits are shifted right. The least significant A accumulator bits are shifted from bit position A_{23} to bit position B_1 of the B accumulator to replace the most significant bits of the B accumulator. The least significant bits of the B accumulator are shifted off bit position B_{23}.



Address Modifiers: None

Timing: The time required to complete this instruction varies with the number of programmed shifts as follows:

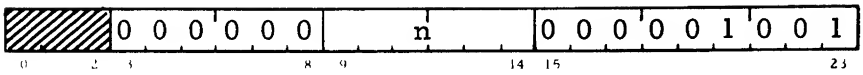
1 - 4	Shifts	- 2 Cycles
5 - 8	Shifts	- 3 Cycles
9 - 12	Shifts	- 4 Cycles
13 - 16	Shifts	- 5 Cycles
17 - 20	Shifts	- 6 Cycles
21 - 23	Shifts	- 7 Cycles

Registers Affected: A accumulator, B accumulator

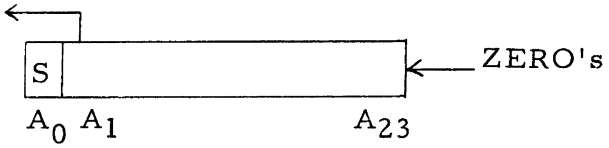
Indicators: None

LSA Left Shift A Accumulator -011

Word Format:



Description: Bits A_1 through A_{23} of the A accumulator are shifted left n places as specified by bits 9 through 14 of the instruction word. The sign bit is unchanged and the most significant bits of the A accumulator are shifted off A_1. The least significant bits are replaced by ZERO's entered into bit A_{23}.



Address Modifiers: None

Timing: The time required to complete this instruction varies with the number of programmed shifts as follows:

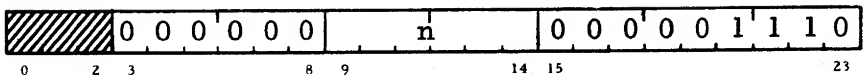
1 - 4	Shifts -	2 Cycles
5 - 8	Shifts -	3 Cycles
9 - 12	Shifts -	4 Cycles
13 - 16	Shifts -	5 Cycles
17 - 20	Shifts -	6 Cycles
21 - 23	Shifts -	7 Cycles

Registers Affected: A accumulator

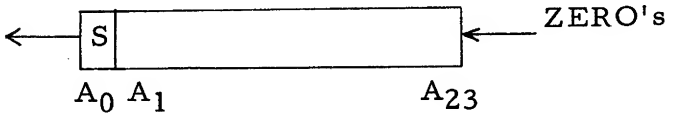
Indicators: None

LSL Left Shift A Accumulator, Logical -016

Word Format:



Description: Bits A_0 through A_{23} of the A accumulator are shifted left n places as specified by bits 9 through 14 of the instruction word. ZERO's are shifted into bit position A_{23} of the A accumulator. The most significant bits (including sign) of the A accumulator are shifted off bit position A_0.



Address Modifiers: None

Timing: The time required to complete this instruction varies with the number of programmed shifts as follows:

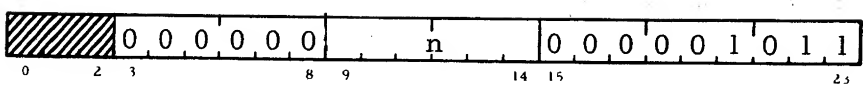
1 - 4	Shifts	- 2 Cycles
5 - 8	Shifts	- 3 Cycles
9 - 12	Shifts	- 4 Cycles
13 - 16	Shifts	- 5 Cycles
17 - 20	Shifts	- 6 Cycles
21 - 23	Shifts	- 7 Cycles

Registers Affected: A accumulator

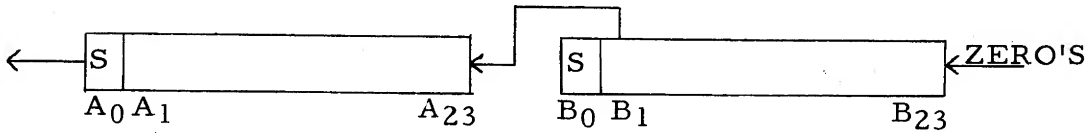
Indicators: None

FLA Full Left Shift Arithmetic -013

Word Format:



Description: Bits A_0 through A_{23} and B_1 through B_{23} of the A and B accumulators are shifted left n places as specified by bits 9 through 14 of the instruction word. ZERO's are entered into bit position B_{23} as the most significant bits of the B accumulator are shifted from bit position B_1 to bit position A_{23} of the A accumulator. The most significant bits of the A accumulator are shifted off bit position A_0.



Address Modifiers: None

Timing: The time required to complete this instruction varies with the number of programmed shifts as follows:

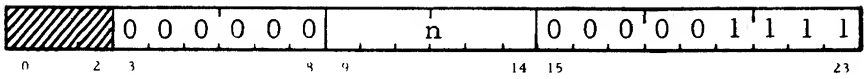
1 - 4	Shifts	- 2 Cycles
5 - 8	Shifts	- 3 Cycles
9 - 12	Shifts	- 4 Cycles
13 - 16	Shifts	- 5 Cycles
17 - 20	Shifts	- 6 Cycles
21 - 23	Shifts	- 7 Cycles

Registers Affected: A accumulator, B accumulator

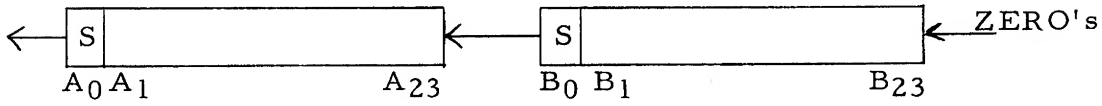
Indicators: None

FLL Full Left Logical -017

Word Format:



Description: Bits A_0 through A_{23} and bits B_0 through B_{23} of the A and B accumulators are shifted left n places as specified by bits 9 through 14 of the instruction word. The sign bits of both accumulators are shifted and the most significant bits of the B accumulator are shifted from bit position B_0 to bit position A_{23} of the A accumulator. The most significant bits of the A accumulator are shifted off bit position A_0 while ZERO's replace the least significant bits of the B accumulator.



Address Modifiers: None

Timing: The time required to complete this instruction varies with the number of programmed shifts as follows:

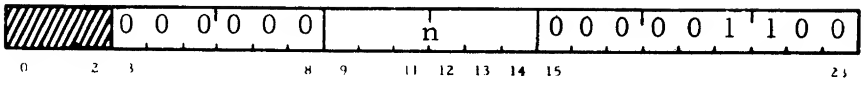
1	- 4	Shifts	- 2 Cycles
5	- 8	Shifts	- 3 Cycles
9	- 12	Shifts	- 4 Cycles
13	- 16	Shifts	- 5 Cycles
17	- 20	Shifts	- 6 Cycles
21	- 23	Shifts	- 7 Cycles

Registers Affected: A accumulator, B accumulator

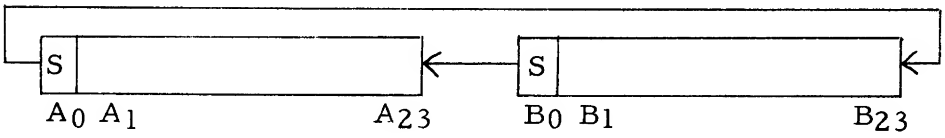
Indicators: None

FRL Full Left Rotate Accumulators **-014**

Word Format:



Description: Bits A_0 through A_{23} and B_0 through B_{23} of the A and B accumulators are rotated left n places as specified by bit 9 through 14 of the instruction word. The sign bit in the A accumulator is shifted to bit position B_{23} of the B accumulator and the sign bit of the B accumulator is shifted to bit position A_{23} of the A accumulator.



Address Modifiers: None

Timing: The time required to complete this instruction varies with the number of programmed shifts as follows:

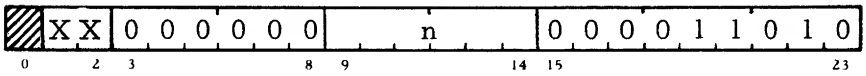
1 - 4	Shifts	- 2 Cycles
5 - 8	Shifts	- 3 Cycles
9 - 12	Shifts	- 4 Cycles
13 - 16	Shifts	- 5 Cycles
17 - 20	Shifts	- 6 Cycles
21 - 23	Shifts	- 7 Cycles

Registers Affected: A accumulator, B accumulator

Indicators: None

NOR Normalize Accumulators -032

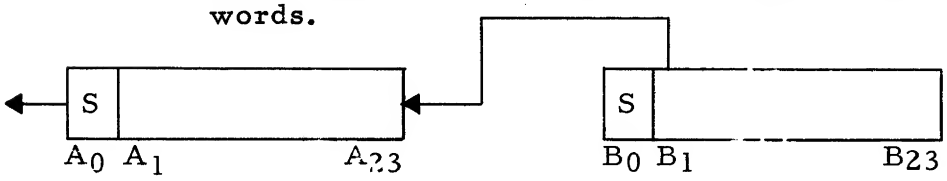
Word Format:



Description: Bits A_0 through A_{23} and B_1 through B_{23} of the A and B accumulators are shifted left until the bit occupying bit position A_1 differs from the bit occupying bit position A_0. ZERO's are shifted into bit position B_{23} of the B accumulator as the most significant bits of that register are shifted from bit position B_1 to bit position A_{23} of the A accumulator. The most significant bits of the A accumulator are shifted off A_0.

Note: Bits 9 through 14 are coded to specify the maximum number of permissible shifts.

This instruction removes the non-significant bits from single or double precision negative and positive words.



Address Modifiers: Index 1 only. If index register 1 is selected by a bit located in bit position 2 of the instruction word, that index register is decremented each time the registers are shifted. This allows creation of the binary exponent in the index register.

Timing: The time required to complete this instruction varies with the number of shifts necessary to normalize the accumulator word as follows:

1 - 4 Shifts - 2 Cycles	24 - 27 Shifts - 8 Cycles
5 - 8 Shifts - 3 Cycles	28 - 31 Shifts - 9 Cycles
9 - 12 Shifts - 4 Cycles	32 - 35 Shifts -10 Cycles
13 - 16 Shifts - 5 Cycles	36 - 29 Shifts -11 Cycles
17 - 20 Shifts - 6 Cycles	40 - 43 Shifts -12 Cycles
21 - 23 Shifts - 7 Cycles	44 - 45 Shifts -13 Cycles

Registers Affected: A accumulator, B accumulator

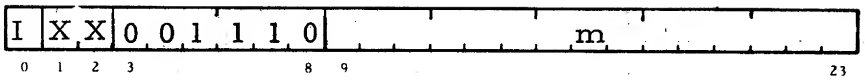
Indicators: None

Control Instructions

The five instructions in this group are used for general "house keeping" functions required by the program. The HLT (halt) instruction stops the computer after loading the next sequential instruction into the instruction register. The NOP (no operation) instruction performs no other function than to reserve a program slot for a future addition or to delay the program to match a real-time input or output rate. The PIE and PID (priority interrupt enable/disable) allow program control of the priority interrupt enables. The EXU (execute) instruction allows the program to execute an instruction out of the normal program sequence without changing the program counter.

EXU Execute 16

Word Format:



Description: The instruction located at the effective memory address is executed.

Note: This instruction allows the use of instructions out of the normal program counter sequence without changing the program count itself. In addition, the effective address may be modified through the index and indirect flags so that different instructions are executed each time the same EXU instructions occur in a program loop.

Address Modifiers: Indirect and Index (1, 2 or 3)

Timing: 1 Cycle plus the time required for the executed instruction

Registers Affected: Dependent on the executed instructions

Indicators: None

HLT Halt -000

Word Format:

Description: Halts the operation of the computer.

Note: The computer stops with the address of the next instruction in the program counter. The START switch is closed to initiate the I cycle of that addressed instruction. The program counter may be manually reset and set to a new starting address prior to the closing of the START switch.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: None

Indicators: HALT

NOP No Operation -022

Word Format:

Description: No operation is performed.

Note: This instruction is used to reserve memory locations for instructions to be added within the program encompassing that memory location. It may also be used to delay a program to match a real-time input or output transfer rate.

Address Modifiers: None

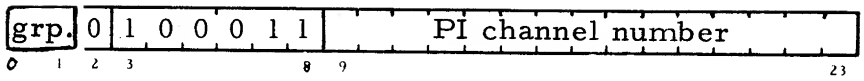
Timing: 1 Cycle

Registers Affected: None

Indicators: None

PID Priority Interrupt Disable **43.0**

Word Format:



Description: Disables a specific priority interrupt channel for subsequent operation. Bits 23 through 9 are set to ONE's to disable interrupt channels 1 through 15 of the specified interrupt group.

BIT LOCATION	FUNCTION
Bits 0 & 1	Group number ($0\text{-}3_8$)
Bit 2	Disable flag (ZERO)
Bits 3 - 8	Op code
Bits 9 - 23	Channel number (15-1)

Note: This instruction allows any or all of the 15 priority interrupt channels belonging to the specified interrupt group to be disabled. The disabled channel will not sense any interrupt signals until again enabled by the PIE instruction.

Address Modifiers: None

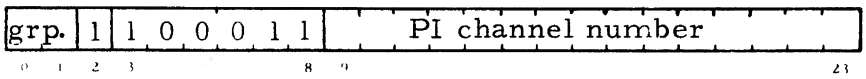
Timing: 1 Cycle

Registers Affected: None

Indicators: None

PIE — Priority Interrupt Enable — 43.1

Word Format:



Description: Enables a specific priority interrupt channel for subsequent operation. Bits 23 through 9 are set to ONE's to enable priority interrupt channels 1 through 15 of the specified interrupt group.

BIT LOCATION	FUNCTION
Bits 0 & 1	Group number ($0\text{-}3_8$)
Bit 2	Enable flag (ONE)
Bits 3 - 8	Op code
Bits 9 - 23	Channel number (15-1)

Note: This instruction allows any or all of the 15 priority interrupt channels in the specified interrupt group to be enabled. The enabled channel will then sense any interrupt signals until disabled by a PID instruction.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: None

Indicators: None

Input/Output Instructions

This group includes eight augmented 17_8 operation code words and three augmented 13_8 operation code words. These instructions are used to communicate with the I/O channels and units. Two of the instructions consist of two memory words and one, the BTC (block transfer) instruction, contains three words. Each extra word is read from memory as a normal part of the instruction execution and requires no additional programming. The MIC, AIC (input from channel), MOC and AOC (output to channel) are all skip instructions. If a channel is ready to receive or transmit data when a transfer instruction not containing a wait flag is issued, the next instruction is skipped. If the channel is not ready, the next instruction is executed. If the transfer instructions contain a wait flag, the computer pauses and the I/O WAIT indicator lights, until the instruction is executed. If an I/O unit is not operating properly, the WAIT indicator alerts the operator to the condition. Four memory reference instructions are provided to allow direct transfers between basic channels A and B and specified memory locations.

The TUN (test unit), TCN (test channel) and TEX (test external) instructions are also skip instructions designed to execute the next instructions if the tested-for condition is absent and to skip that instruction if the condition is present. The SUN (select unit), SCN (select channel) and ACT (select external) instructions contain an optional wait flag that delays the program counter advance until an "Instruction Received" signal is received from the channel or unit. The delay is provided only to allow for propagation delay on long lines and in no way tests the unit or channel for a ready condition and, if the channel or unit is not ready, the select instruction will be ignored. For this reason, the channel or unit should first be tested, then selected. If the wait flags used and the channel or unit malfunctions so that no "Instruction Received" signal is generated, the computer will halt.

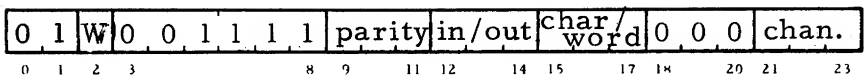
The word formats of these augmented codes vary somewhat from instruction to instruction. The assigned significance of each bit is described for each word. All unassigned bits should be set to zero whenever the instructions are entered into the computer.

SCN

Select Channel (n)

17.20 (No Wait Flag)
17.30 (Wait Flag)

Word Format:



Description: Selects and commands channel (n) to perform a specific operation.

BIT LOCATION	FUNCTION
Bits 0 & 1	Augment code
Bit 2	Wait flag
Bits 3 - 8	Op code
Bits 9 - 11	Parity select (000 - even) (001 - odd)
Bits 12 - 14	In/Out (000 - output) (001 - input)
Bits 15 - 17	Character/word ($0-3_8$)
Bits 18 - 20	Augment code
Bits 21 - 23	Channel number ($0-7_8$)

Note: If the wait flag is set (ONE), the program counter advance is inhibited until an "Instruction Received" signal is returned by the channel. This delay is provided only to allow for propagation delay on long I/O lines and in no way tests the channel for a ready condition. If the channel is not ready, the select instruction will be ignored. For this reason, this instruction should be preceded by a test instruction when in doubt of the ready status.

Address Modifiers: None

Timing: 1 or 2 cycles depending on the propagation delay of the "Instruction Received" return signal.

Registers Affected: None

Indicators: I/O WAIT

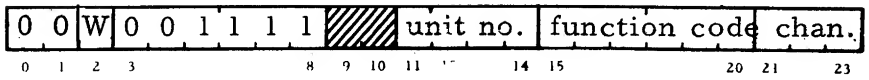
Mnembler Format: SCN Chan. No, In/out, chars/word, Wait, Parity

SUN

Select Unit (n)

17.0 (No Wait Flag)
17.1 (Wait Flag)

Word Format:



Description: Selects and commands input/output unit (n) to perform a specific operation.

BIT LOCATION	FUNCTION
Bits 0 & 1	Augment code
Bit 2	Wait flag
Bits 3 - 8	Op code
Bits 9 - 10	Unassigned
Bits 11 - 14	Unit number ($00-17_8$)
Bits 15 - 20	Function code ($00-77_8$)
Bits 21 - 23	Channel number ($0-7_8$)

Note: If the wait flag is set (ONE), the program counter advance is inhibited until an "Instruction Received" signal is returned by the unit. This delay is provided only to allow for propagation delay on long I/O lines and in no way tests the unit for a ready condition. If the unit is not ready, the select instruction will be ignored. For this reason, this instruction should be preceded by a test instruction when in doubt of the ready status.

Address Modifiers: None

Timing: 1 or 2 cycles depending on the propagation delay of the "Instruction Received" return signal.

Registers Affected: None

Indicators: I/O WAIT

Mnembler Format: SUN Chan. No. , Unit No, Function code, Wait

AOC

Accumulator Word Out to Channel (n) **17.21** (No Wait Flag) **17.31** (Wait Flag)

Word Format:

Description: Transfers the contents of the A accumulator to channel (n). The contents of the A accumulator are unchanged.

BIT LOCATION	FUNCTION
Bits 0 & 1	Augment code
Bit 2	Wait flag
Bits 3 - 8	Op code
Bits 9 - 17	Unassigned
Bits 18 - 20	Augment code
Bits 21 - 23	Channel number ($0-7_8$)

Note: If the wait flag is set (ONE), the computer will pause if the channel is not ready and resume when when the channel becomes ready.

If the wait flag is not set (ZERO), the computer will skip the next instruction if the channel is ready. If the channel is not ready, the next instruction will be executed. This skip function allows the program to continue with unrelated processing and returns to the AOC instruction rather than merely wait.

Address Modifiers: None

Timing: 1 Cycle and wait

Registers Affected: Channel (n) buffer

Indicators: I/O WAIT

Mnembler Format: AOC Chan. No., Wait

AIC Accumulator Word in From Channel (n) **17.22** (No Wait Flag) **17.32** (Wait Flag)

Word Format:

Description: Transfers the contents of channel (n) to the A accumulator. The contents of channel (n) are unchanged.

BIT LOCATION	FUNCTION
Bits 0 & 1	Augment code
Bit 2	Wait flag
Bits 3 - 8	Op code
Bits 9 - 17	Unassigned
Bits 18 - 20	Augment code
Bits 21 - 23	Channel number ($0-7_8$)

Note: If the wait flag is set (ONE), the computer will pause if the channel is not ready and resume when when the channel becomes ready.

If the wait flag is not set (ZERO), the computer will skip the next instruction if the channel is ready. If the channel is not ready, the next instruction will be executed. This skip function allows the program to continue with unrelated processing and returns to the AIC instruction rather than merely wait.

Address Modifiers: None

Timing: 1 Cycle and wait

Registers Affected: A accumulator

Indicators: I/O WAIT

Mnembler Format: AIC Chan. No., Wait

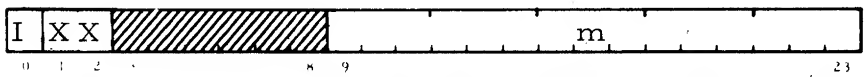
MOC

Output Memory Word to Channel (n)

17.23 (No Wait Flag)
17.33 (Wait Flag)

Word Format:

Word 1



Word 2

Description: The contents of the effective memory address are transferred to the channel (n) buffer. The contents of the memory are unchanged.

	BIT LOCATION	FUNCTION
Word 1	Bits 0 & 1	Augment code
	Bit 2	Wait flag
	Bits 3 - 8	Op code
	Bits 9 - 17	Unassigned
	Bits 18 - 20	Augment code
	Bits 21 - 23	Channel number
Word 2	Bit 0	Indirect flag
	Bits 1 & 2	Index register selectors
	Bits 3 - 8	Unassigned
	Bits 9 - 23	Memory address ($00000-77777_8$)

Note: If the wait flag is set (ONE), the computer will pause if the channel is not ready and resume when the channel becomes ready.

If the wait flag is not set (ZERO), the computer will skip the next instruction if the channel is ready. If the channel is not ready, the next instruction will be executed. This skip function allows the program to continue with unrelated processing and return to the MOC instruction rather than merely wait.

Address Modifiers: Indirect and Index (1, 2 or 3)

Timing: 3 Cycles and wait

Registers Affected: Channel (n) buffer

Indicators: I/O WAIT

Mnembler Format: MOC Chan. No., Wait
DAC* Addr, Index

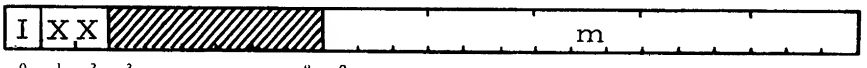
MIC

Input Memory Word from Channel (n)

17.24 (No Wait Flag)
17.34 (Wait Flag)

Word Format:

Word 1



Word 2

Description: Transfers the contents of the channel (n) buffer to the effective memory address. The contents of the channel (n) buffer are unchanged.

	BIT LOCATION	FUNCTION
Word 1	Bits 0 & 1	Augment code
	Bit 2	Wait flag
	Bits 3 - 8	Op code
	Bits 9 - 17	Unassigned
	Bits 18 - 20	Augment code
	Bits 21 - 23	Channel number ($0-7_8$)
Word 2	Bit 0	Indirect flag
	Bits 1 & 2	Index register selectors
	Bits 3 - 8	Unassigned
	Bits 9 - 23	Memory address

Note: If the wait flag is set (ONE), the computer will pause if the channel is not ready and resume when the channel becomes ready.

If the wait flag is not set (ZERO), the computer will skip the next instruction if the channel is ready. If the channel is not ready, the next instruction will be executed. This skip function allows the program to continue with unrelated processing and return to the MIC instruction rather than merely wait.

Address Modifiers: Indirect and Index (1, 2 or 3)

Timing: 3 Cycles and wait

Registers Affected: None

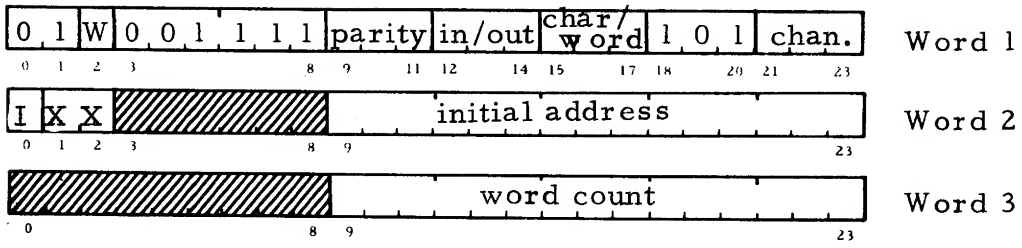
Indicators: I/O WAIT

Mnembler Format:
MOC Chan. No, Wait
DAC* Addr, Index

BTC

Block Transfer to Channel (n) **17.25** (No Wait Flag)
17.35 (Wait Flag)

Word Format:



Description: Initiates the transfer of a block of words between successive memory addresses and the channel (n) buffer. The first (lowest) memory address is contained in the second instruction word and the word count (words/block) is contained in the third word.

	BIT LOCATION	FUNCTION
Word 1	Bits 0 & 1	Augment code
	Bit 2	Wait flag
	Bits 3 - 8	Op code
	Bits 9 - 11	Parity selector (000 - even) (001 - odd)
	Bits 12 - 14	In/out (000 - output) (001 - input)
	Bits 15 - 17	Character/word ($0-3_8$)
	Bits 18 - 20	Augment code
	Bits 21 - 23	Channel number ($0-7_8$)
Word 2	Bit 0	Indirect flag
	Bits 1 & 2	Index register selectors
	Bits 3 - 8	Unassigned
	Bits 9 - 23	Initial memory address ($00000-77776_8$)
Word 3	Bits 0 - 8	Unassigned
	Bits 9 - 23	Word count (00001-777778)

Note: The block transfer initiated by this instruction operates on a time-share basis with the main frame program, having memory access priority over the program. If a block transfer is in process and a priority interrupt occurs, the priority interrupt program supplants the main frame program and then shares the memory with the block transfer.

Address Modifiers: Indirect and Index (1, 2 or 3)

Timing: 3 Cycles and wait

Registers Affected: channel (n) buffer

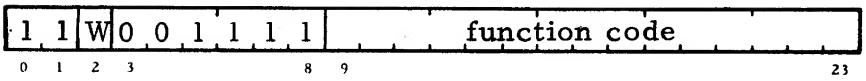
Indicators: I/O WAIT

Mnembler Format:
BTC Chan. No., In/out, chars/word, Wait, Parity
DAC* Addr, Index
DATA Word count (5 octal digits)

ACT

Activate, Command and Transfer **17.6** (No Wait Flag)
17.7 (Wait Flag)

Word Format:



Description: A general-purpose input/output instruction used to initiate a specific action by a none-standard input/output unit.

BIT LOCATION	FUNCTION
Bits 0 & 1	Augment code
Bit 2	Wait flag
Bits 3 - 8	Op code
Bits 9 - 23	Function code ($00000-77777_8$) or (1-15) as required.

Note: If the wait flag is set (ONE), the program counter advance is inhibited until an "Instruction Received" signal is returned by the unit. This delay is provided only to allow for propagation delay on long I/O lines and in no way tests the unit for a ready condition. If the unit is not ready, the select instruction will be ignored. For this reason, this instruction should be preceded by a test instruction when in doubt of the ready status.

Address Modifiers: None

Timing: 1 or 2 cycles depending on the propagation delay of the "Instruction Received" return signal.

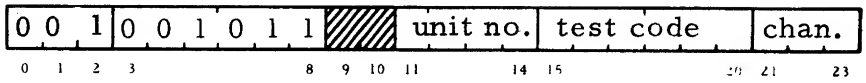
Registers Affected: None

Indicators: I/O WAIT

Mnembler Format: Function code, Wait

TUN Test Unit (n) 13.1

Word Format:



Description: Tests unit (n) of channel (n) for the presence of a specific condition. If the condition is present, the next instruction (NI) is executed; if the condition is absent, the second sequential instruction (NI+1) is executed.

BIT LOCATION	FUNCTION
Bits 0 & 1	Augment code
Bit 2	Wait flag
Bits 3 - 8	Op code
Bits 9 - 10	Unassigned
Bits 11 - 14	Unit number ($00-17_8$)
Bits 15 - 20	Test code ($00-77_8$)
Bits 21 - 23	Channel number ($0-7_8$)

Address Modifiers: None

Timing: 1 or 2 cycles depending on propagation delay of the Instruction Received return signal.

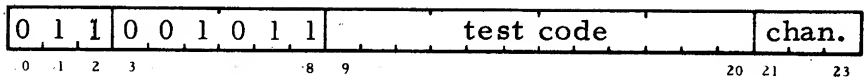
Registers Affected: Program counter

Indicators: I/O WAIT

Mnembler Format: Chan. No., Unit No., Test code

TCN Test Channel (n) 13.3

Word Format:



Description: Tests channel (n) for the presence of a specific condition. If the condition is present, the next instruction (NI) is executed; if the condition is absent, the second sequential instruction (NI+1) is executed.

BIT LOCATION	FUNCTION
Bits 0 & 1	Augment code
Bit 2	Wait flag
Bits 3 - 8	Op code
Bits 9 - 20	Test code ($0000-7777_8$)
Bits 21 - 23	Channel number ($0-7_8$)

Address Modifiers: None

Timing: 1 or 2 cycles depending on the propagation delay of the Instruction Received return signal.

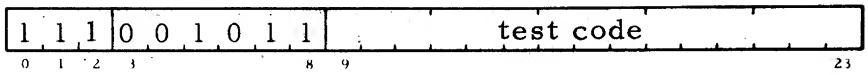
Registers Affected: Program counter

Indicators: I/O WAIT

Mnembler Format: TCN Chan. No. , Test code

TEX Test External Device 13.7

Word Format:



Description: A general-purpose input/output instruction used to test for specific conditions in non-standard peripheral equipment.

BIT LOCATION	FUNCTION
Bits 0 & 1	Augment code
Bit 2	Wait flag
Bits 3 - 8	Op code
Bits 9 - 23	Test code ($00000-77777_8$) or (1-15) as required.

Address Modifiers: None

Timing: 1 or 2 cycles depending on the propagation delay time of the "Instruction Received" return signal.

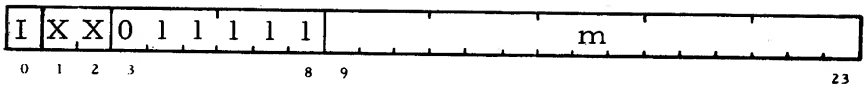
Registers Affected: None

Indicators: I/O WAIT

Mnembler Format: TEX Test code

MCA Memory to Channel A 37

Word Format:



Description: Transfers the contents of the effective memory address to Channel A. The contents of the memory are unchanged.

Note: Since this instruction does not contain a wait flag, it must be used only when the channel is known to be ready.

Address Modifiers: Indirect and Index (1, 2 or 3)

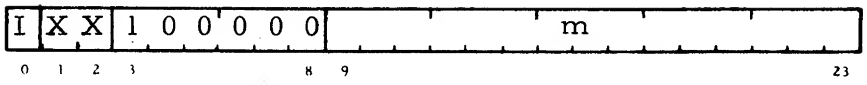
Timing: 2 Cycles

Registers Affected: Channel A buffer

Indicators: None

MCB Memory to Channel B 40

Word Format:



Description: The contents of the effective memory address are transferred to Channel B. The contents of the memory are unchanged.

Note: Since this instruction does not contain a wait flag, it must be used only when the channel is known to be ready.

Address Modifiers: Indirect and Index (1, 2 or 3)

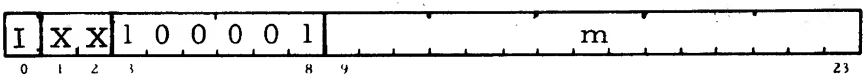
Timing: 2 Cycles

Registers Affected: Channel B

Indicators: None

CAM Channel A to Memory 41

Word Format:



Description: The contents of Channel A are transferred to the effective memory address. The contents of Channel A are unchanged.

Note: Since this instruction does not contain a wait flag, it must be used only when the channel is known to be ready.

Address Modifiers: Indirect and Index (1, 2 or 3)

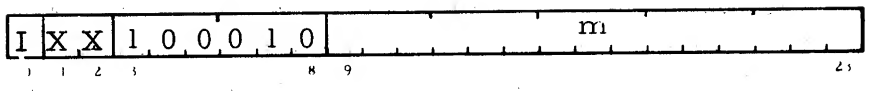
Timing: 2 Cycles

Registers Affected: None

Indicators: None

CBM Channel B to Memory 42

Word Format:



Description: The contents of Channel B are transferred to the effective memory address. The contents of Channel B are unchanged.

Note: Since this instruction does not contain a wait flag, it must be used only when the channel is known to be ready.

Address Modifiers: Indirect and Index (1, 2 or 3)

Timing: 2 Cycles

Registers Affected: None

Indicators: None

EAU Instructions

The EAU instructions apply only to the optional extended arithmetic unit. Four of the instructions are used to set the operating mode to single precision fixed point, double precision fixed point, normalized floating point or un-normalized floating point. Four others allow selection of either add, subtract, multiply or divide operations. Three instructions are used to load normal, load negated, or store the contents of the 48-bit EA accumulator. Two instructions allow the loading and storing of the least significant half of the EA accumulator. The remaining instructions include instructions to transfer words between the EA and EB accumulators, normalizing the EA accumulator and skip instructions that test the EAU ready state, the contents of the EA accumulator and arithmetic overflow.

Word Format:

Description: Sets the extended arithmetic unit to the floating point mode. All subsequent inputs will be treated as floating point data and all subsequent instructions will operate in the floating point mode.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: None

Indicators: None

Word Format:

Description: Sets the extended arithmetic unit to the single-precision mode. All subsequent inputs will be treated as single-precision data and all subsequent instructions will operate in the single-precision mode.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: None

Indicators: None

Word Format:

Description: Sets the extended arithmetic unit to the double-precision mode. All subsequent inputs will be treated as double-precision data and all subsequent instructions will operate in the double-precision mode.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: None

Indicators: None

EFU Extended Floating Point, Un-normalized 21.11

Word Format:

Description: Sets the extended arithmetic unit to the un-normalized floating point mode. The results of all arithmetic operations remain un-normalized in this mode.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: None

Indicators: None

ENO Extended Normalize 21.00

Word Format:

Description: The contents of the EA accumulator together with the contents of the EB accumulator are shifted left until the most significant bit of the EA accumulator differs from the sign bit of the EA accumulator. The 39 most significant bits of the EA accumulator then form the mantissa of the normalized word. The number of shifts required to normalize the contents of the accumulators is counted by the 9 bit exponent which then holds the exponent of the resulting mantissa at the conclusion of the normalize operation.

Address Modifiers: None

Timing: 1 Cycle plus 1 cycle for each pair required shifts

Registers Affected: EA accumulator, EB accumulator, Exponent register

Indicators: None

EIA Interchange EA Accumulator and EB Accumulator 21.01

Word Format:

Description: The contents of the EA accumulator replace the previous contents of the EB accumulator and the contents of the EB accumulator replace the previous contents of the EA accumulator.

Address Modifiers: None

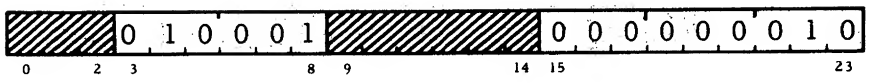
Timing: 1 Cycle

Registers Affected: EA accumulator, EB accumulator

Indicators: None

EBA Transfer EB Accumulator to EA Accumulator 21.02

Word Format:



Description: The contents of the EB accumulator replace the previous contents of the EA accumulator. The contents of the EB accumulator are unchanged.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: EA accumulator

Indicators: None

EAB Transfer EA Accumulator to EB Accumulator 21.03

Word Format:

Description: The contents of the EA accumulator replace the previous contents of the EB accumulator. The contents of the EA accumulator are unchanged.

Address Modifiers: None

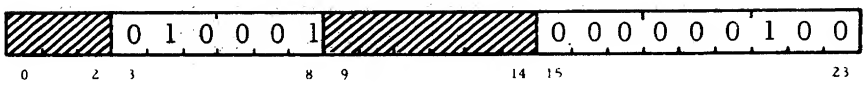
Timing: 1 Cycle

Registers Affected: EB accumulator

Indicators: None

ESR Extended Skip if Ready **21.04**

Word Format:



Description: If the EAU is not executing a previous instruction and is operative, the next instruction (NI) is skipped and the second sequential instruction (NI+1) is executed. If the EAU is active or inoperative, the next instruction (NI) is executed.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: Program counter

Indicators: None

ESZ Extended Skip if Zero 21.05

Word Format:

Description: If the contents of the EA accumulator equal zero, the next instruction (NI) is skipped and the second sequential instruction (NI+1) is executed. If the contents of the EA accumulator are greater than or less than zero, the next instruction (NI) is executed.

Address Modifiers: None

Timing: 1 Cycle

Registers Affected: Program counter

Indicators: None

EPS Extended Skip if Positive **21.06**

Word Format:

Description: If the contents of the EA accumulator are positive, the next instruction (NI) is skipped and the second sequential instuction (NI+1) is executed. If the contents of the EA accumulator are negative, the next instruction (NI) is executed.

Address Modifiers: None

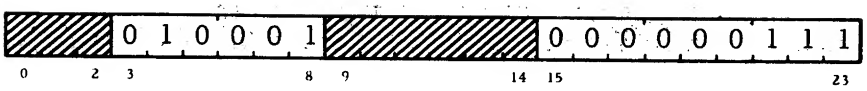
Timing: 1 Cycle

Registers Affected: Program counter

Indicators: None

ESN Extended Skip if Negative 21.07

Word Format:



Description: If the contents of the EA accumulator are negative, the next instruction (NI) is skipped and the second sequential instruction (NI+1) is executed. If the contents of the EA accumulator are equal to zero or positive, the next instruction (NI) is executed.

Address Modifiers: None

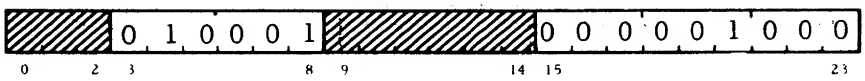
Timing: 1 Cycle

Registers Affected: Program counter

Indicators: None

ESO Extended Skip on Overflow 21.10

Word Format:



Description: If the EAU overflow latch has been set, the next instruction (NI) is skipped, the second sequential instruction (NI+1) is executed and the overflow latch is reset. If the overflow latch is reset, the next sequential instruction (NI) is executed.

Address Modifiers: None

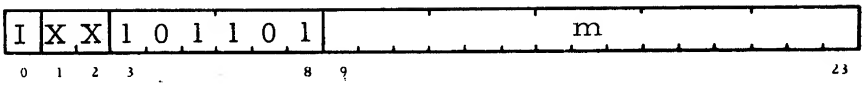
Timing: 1 Cycle

Registers Affected: Program counter

Indicators: None

ESL Store Least Significant Half of EA Accumulator 55

Word Format:



Description: The contents of the 24 least significant bits of the EA accumulator replace the previous contents of the effective memory address.

Note: This instruction is effective only in the fixed point single-precision mode and is treated as a NOP in the other modes.

Address Modifiers: Indirect and Index (1, 2 or 3)

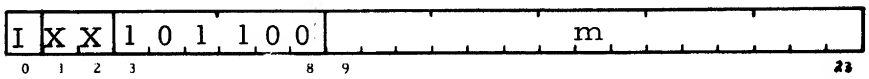
Timing: 2 Cycles

Registers Affected: None

Indicators: None

ELL Load Least Significant Half EA Accumulator 54

Word Format:



Description: The contents of the effective memory address replace the previous contents of the 24 least significant bits of the EA accumulator and clears the 24 most significant bit positions to zeros.

Note: This instruction is effective only in the fixed point single-precision mode and is treated as a NOP in the other modes.

Address Modifiers: Indirect and Index (1, 2 or 3)

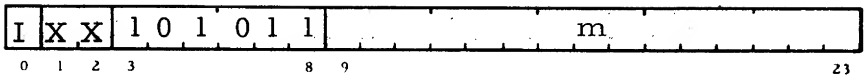
Timing: 2 Cycles

Registers Affected: EA accumulator

Indicators: None

EST Extended Store 53

Word Format:



Description:

In the fixed point single-precision mode, the 24 most significant bits of the EA accumulator replace the previous contents of the effective memory address.

In the fixed point double-precision mode, the contents of the EA accumulator replace the previous contents of the effective memory address and the next sequential memory address. The most significant 24 bits of the EA accumulator are transferred to the effective memory address and the 24 least significant bits are transferred to the second memory word.

In the floating point mode, the contents of the EA accumulator and the exponent register replace the previous contents of the effective memory address and the next sequential memory address. The 24 most significant bits of the EA accumulator are transferred to the effective memory address, the 15 next most significant bits of the EA accumulator replace the 15 most significant bits of the second memory word and the 9 bit contents of the exponent register replace the 9 least significant bits of the second memory address.

Address Modifiers: Indirect and Index (1, 2 or 3)

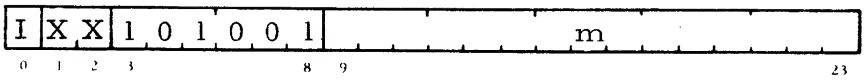
Timing:
2 Cycles in single-precision
3 Cycles in double-precision and floating point

Registers Affected: None

Indicators: None

ELN Extended Load Negative 51

Word Format:



Description:

In the fixed point single-precision mode, the contents of the effective memory address are two's complemented and replace the 24 most significant bits of the EA accumulator. The 24 least significant bits of the EA accumulator are cleared to zeros.

In the fixed point double-precision mode, the contents of the effective memory address and the next sequential memory address are two's complemented and replace the previous contents of the EA accumulator.

In the floating point mode, the contents of the effective memory address and the next sequential memory address are complemented and replace the previous contents of the 39 most significant bits of the EA accumulator. The least significant 9 bits of the second memory word replace the previous contents of the exponent register and the 9 least significant bits of the EA accumulator are cleared to zeros.

Address Modifiers: Indirect and Index (1, 2 or 3)

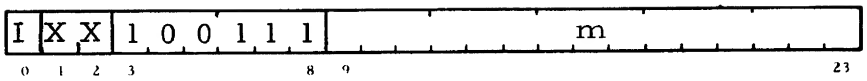
Timing:
2 Cycles in single-precision
3 Cycles in double-precision and floating point

Registers Affected: EA accumulator, Exponent register

Indicators: None

EMU Extended Multiply 47

Word Format:



Description:

In the fixed point single-precision mode, the contents of the effective memory address are multiplied by the least significant 24 bits of the EA accumulator. The product is stored in the EA accumulator.

In the fixed point double-precision mode, the contents of the effective memory address and the contents of the next sequential memory address are multiplied by the contents of EB accumulator. The most significant half of the product is stored in the EA accumulator and the least significant half of the product is stored in the EB accumulator.

In the floating point mode, the 38 most significant bits of the double-length contents of the effective memory address and the next sequential memory address are multiplied by the 38 most significant bits of the EB accumulator. The 9 bit exponents from the second memory word (multiplicand) and from the exponent register (multiplier) are added. The product mantissa is stored in the 39 most significant bits of the EA accumulator and the product exponent is stored in the exponent register.

Address Modifiers: Indirect and Index (1. 2 or 3)

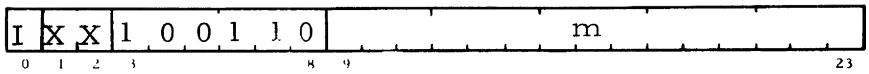
Timing: 15 Cycles in single-precision mode,
30 Cycles in double-precision,
23 Cycles plus 1 cycle for pair of shifts to renormalize in the flowing point mode.

Registers Affected: EA accumulator, EB accumulator, Exponent register

Indicators: OVERFLOW if the exponent sum exceeds 8 bits plus sign in the floating point mode only.

ESU Extended Arithmetic Subtract 46

Word Format:



Description:

In the fixed point single-precision mode, the contents of the effective memory address are algebraically subtracted from the 24 most significant bits of the EA accumulator. The resulting difference is stored in the 24 most significant bits of the EA accumulator.

In the fixed point double-precision mode, the contents of the effective memory address and the contents of the next sequential memory address are algebraically subtracted from the contents of the EA accumulator. The resulting difference is stored in the EA accumulator.

In the floating point mode, the minuend and subtrahend exponents are compared after the subtrahend is negated. If they differ by more than 38 bits (the size of the mantissa), the larger operand is placed, or remains, in the EA accumulator and the instruction is terminated. If the exponents differ by less than 38 bits, the smaller operand is decreased until its exponent equals the exponent of the larger operand and the mantissas are then algebraically subtracted. The mantissa of the resulting difference is stored in the 39 most significant bit positions of the EA accumulator. The exponent remains in the exponent register.

Note: The minuend must be properly positioned in the EA accumulator prior to the ESU instruction. This may be accomplished by an ELO instruction or as a result of a preceding arithmetic operation.

Address Modifiers: Indirect and Index (1, 2 or 3)

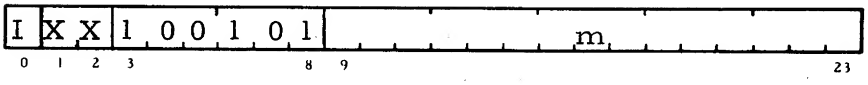
Timing:
2 Cycles in single-precision
3 Cycles in double-precision
3 Cycles plus 1 cycle for each pair of shifts to renormalize and to shift exponents in floating point mode.

Registers Affected: EA accumulator and Exponent registers

Indicators: OVERFLOW if the difference exceeds 23 bits plus sign in single-precision, 46 bits plus sign in double-precision, or if the exponent exceeds 8 bits plus sign in the floating point mode.

EAD Extended Arithmetic Add 45

Word Format:



Description:

In the fixed point single-precision mode, the contents of the effective memory address are algebraically added to the 24 most significant bits of the EA accumulator. The resulting sum is stored in the 24 most significant bits of the EA accumulator.

In the fixed point double-precision mode, the contents of the effective memory address and the contents of the next sequential memory address are added to the total contents of the EA accumulator. The resulting sum is stored in the EA accumulator.

In the floating point mode, the augend and addend exponents are compared. If they differ by more than 38 bits (the size of the mantissa), the larger operand is placed, or remains, in the EA accumulator and the instruction is terminated. If the exponents differ by less than 38 bits, the smaller operand is decreased until its exponent equals the exponent of the larger operand and the mantissas are then algebraically added. The mantissa of the resulting sum is stored in the 39 most significant bit positions of the EA accumulator. The exponent remains in the exponent register.

Note: The augend must be properly positioned in the EA accumulator prior to the EAD instruction. This may be accomplished by an ELO instruction or as the result of a preceding arithmetic operation.

Address Modifiers: Indirect and Index (1, 2 or 3)

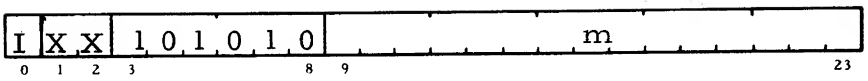
Timing: 2 Cycles in single-precision
3 Cycles in double-precision
13 Cycles plus 1 cycle for each pair of shifts to renormalize and to shift exponents in floating point mode.

Registers Affected: EA accumulator, Exponent registers

Indicators: OVERFLOW if the sum exceeds 23 bits plus sign in single-precision, 46 bits plus sign in double-precision or if the exponent exceeds 8 bits plus sign in the floating point mode.

ELO Extended Load 52

Word Format:



Description:

In the fixed point single-precision mode, the contents of the effective memory address replace the previous contents of the 24 most significant bits of the EA accumulator. The least significant 24 bits of the EA accumulator are cleared to zeros.

In the fixed point double-precision mode, the contents of the effective memory address and the next memory address replace the previous contents of the EA accumulator. The first memory word replace the most significant 24 bits of the EA accumulator and the contents of the second memory word replace the 24 least significant bits of the EA accumulator.

In the floating point mode, the 38 bit mantissa is loaded into the 39 most significant bit positions of the EA accumulator and the 9 bit exponent is loaded into the exponent register. The remaining 9 bits of the EA accumulator are cleared to zeros.

Address Modifiers: Indirect and Index (1, 2 or 3)

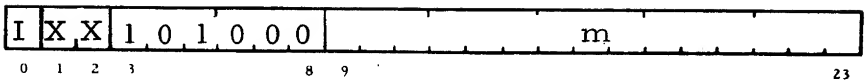
Timing:
2 Cycles in single-precision
3 Cycles in double-precision and floating point

Registers Affected: EA accumulator, Exponent register

Indicators: None

EDV Extended Divide 50

Word Format:



Description:

In the fixed point single-precision mode, the contents of the effective memory address are divided into the contents of the EA accumulator. The quotient is stored in the 24 least significant bits of the EA accumulator and the remainder is stored in the 24 most significant bits of the EA accumulator.

In the fixed point double-precision mode, the contents of the effective memory address and the contents of the next sequential memory address are divided into the 96-bit contents of EA and EB accumulators. The 48-bit quotient is stored in the EA accumulator and the 48-bit remainder is stored in the EB accumulator.

In the floating point mode, the 38 most significant bits of the double length word in the effective memory address and the next sequential memory address are divided into the most significant 38 bits in the EB accumulator and the 8-bit divisor exponent is subtracted from the 8 bit dividend exponent. The quotient mantissa is stored in the 39 most significant bits of the EA accumulator and the quotient exponent is stored in the exponent register.

Address Modifiers: Indirect and Index (1, 2 or 3)

Timing:
17 Cycles in the single-precision mode
32 Cycles in the double-precision
28 Cycles plus 1 cycle for each pair of shifts to renormalize in floating point.

Registers Affected: EA accumulator, EB accumulator, Exponent register

Indicators: OVERFLOW if the quotient exceeds the capacity of the register.

SECTION III SEL 840 SOFTWARE SYSTEM

A comprehensive, fully integrated, well documented and completely checked out program preparation, library, debugging and utility system is supplied with the SEL 840 computer system at no cost to the user.

Specific features of this standard package are listed below by item but a few words in regard to the philosophy behind the software system package design are appropriate here.

In determing the optimum software package for the type of equipment under consideration, the following factors were deemed to be of prime importance.

- The large variety of equipment configurations which will be delivered.

- The type of applications which will be programmed for the equipment.

The amount of programming personnel time which will be involved in developing and debugging operational programs.

- The need to utilize programs and routines which may already exist on other equipment.

- The quality and completeness of the documentation supplied with the software and library routines.

In order to satisfy these objectives, two basic types of program preparation systems are provided; a symbolic assembler (MNEMBLER) and a full FORTRAN IV compiling system. Depending upon the specific requirements of a specific portion of an operational package, these two programming systems provide the user with an optimum capability where trade-offs between coding and checkout time and program running time are involved.

The fact that a specific portion of a program may be coded in either language is most significant to the user. Two methods of mixing are available a.) the FORTRAN IV processor will accept assembler coding 'in-line' or b.) the loader will accept both FORTRAN and assembler generated coding in any sequence.

This feature together with the very comprehensive debug package will significantly reduce the coding and checkout time required to produce operational programs.

The FORTRAN IV language specified for this system is the standard ACM published language which includes the IBM 7090/94 language as a sub-set, thus the SEL FORTRAN IV will provide direct compatibility with the majority of other manufacturer supplied FORTRAN IV systems.

In order to satisfy the requirement that all of the supplied software system will operate on a wide variety of computer configurations, especially in the area of peripheral equipment, all of the supplied packages are written in a modular form with a standard program interface specification. This will allow any combination of input/output equipments to be utilized without major revision of the programs.

840 Assembly Program (MNEMBLER)

The MNEMBLER program operates in either a one-or two-pass mode. The one-pass mode offers a significant time advantage at assembly time, while the two-pass mode results in a shorter output tape and more complete error checking.

All computer instructions will be accepted by the assembler and addresses may be expressed in symbolic, decimal or octal formats.

The following special pseudo-ops are also processed:

BSS	-	Reserve block of storage; name at start.
BES	-	Reserve block of storage; name at end.
EQU	-	Define symbolic name.
ORG	-	Set next storage address.
ZZZ	-	Set instruction bits to zero.
REL	-	Set assembly mode to relative.
ABS	-	Set assembly mode to absolute.
CALL	-	Call library subroutine.
NAME	-	Define subroutine name.
DATA	-	Define octal, decimal (fixed or floating) or Alphanumeric data.
MOR	-	End of tape; more left in program.
END	-	End of program.

A symbolic side-by-side listing complete with error messages is output (operator option) along with the object output tape.

SEL 840 Loader

The SEL 840 object program loader is designed to be compatible with the FORTRAN IV Compiler and the Assembly program.

The program provides for relocatable and absolute instructions.

The capability of using pre-assembled subroutine libraries is included in a manner which guarantees that a given routine will only be loaded once, regardless of the ordering of the library.

Loading and (chaining if requested) is done using modular input/output driver subroutines, allowing maximum flexibility in choice of load device. The loader loads the main program and any subroutines called for. It completes linkage between the main program and the subroutines.

SEL 840 FORTRAN IV

Ease of use was a prime consideration in the design of the compiler. As a result, programmers are free of the restrictions often found in other systems. Convenience features include:

1. One-pass Operation - From source language to machine language object code is a standard feature.

2. No Reserved Identifiers - All names are available for use as identifiers. For example, SEL FORTRAN readily distinguishes that:

   FORMAT (5H) = A(2) is a FORMAT statement and
   FORMAT (H5) = A(2) is an arithmetic statement.

3. Optional In-line Assembly - This feature allows assembly language coding to be intermixed with FORTRAN statements.

4. Optional Tracing - This feature allows selective object code tracing for diagnostic purposes.

5. Optional Mapping - This feature provides a listing of the sub-programs required for execution and the names or values and relative location assignments of all variable/array names and constant values used by the program.

6. Optional Chaining - This feature provides for sequential execution of segmented programs.

SEL 840 Debug

The debug program is a utility program designed to help a programmer debug a program while it is in memory. The following functions are provided:

a. Type the contents of specified memory in octal or command format.

b. Modify the specified memory. Input being in octal format.

c. Dump specified memory areas onto paper tape in a self-loading (non-relocatable) format

d. Enter breakpoints in order to "leap-frog" trace a program.

e. Clear specified areas of memory to zero.

f. Search memory for references to specified areas.

g. Initiate branches (or HALT AND BRANCH) to any part of memory.

Each of these functions are initiated by typing a keyword through the console typewriter keyboard.

SEL 840 UPDATE

Correction of errors in card decks is a relatively easy procedure, consisting of pulling out the bad cards and inserting new cards. However, symbolic source programs on paper tapes are not so easily corrected or modified

The UPDATE program is designed to allow the computer operator to easily correct or modify a symbolic source program tape by providing the following functions:

a. Deletion of a specified line or group of lines.

b. Insertion of a new or replacement line or lines

c. List the source program (or portions of it) complete with line reference numbers.

All references to the symbolic source tape are made by referring to a sequence number. This number is present on all assembly listings and on all listings generated by the UPDATE program.

SEL 840 Library Package

The following subroutines constitute the Library at the time of writing.

Single Precision Floating Point Functions

Double Precision Floating Point Functions

Complex Floating Point Functions

Integer Functions

Input/Output Functions

Control Functions

MNE-MONIC	OP CODE	FUNCTION	PAGE
AAM	31	Add (M) to (A)	2-12
ACT	17.6	Ext. Command	2 70
AIC	17.22	(Chan.) to A	2-66
AMA	05	Add (A) to (M)	2-11
AMX	61	Add (M) to (X)	2-13
AOC	17.21	(A) to chan.	2-65
ASC	-020	Comp. A sign	2-36
BAN	23	Branch if (A) neg.	2-24
BAP	24	Branch if (A) pos.	2-25
BAZ	22	Branch if (A)=0	2-23
BOF	25	Branch on O'FLOW	2-26
BRU	11	Branch to M	2-19
BTC	17.25	Block Transfer	2-69
CAM	41	(Chan. A) to M	2-76
CBM	42	(Chan. B) to M	2-77
CLA	-003	Clear (A) to 0	2-41
CMA	15	Compare (A) and (M)	2-22
CNS	20	Convert No. System	2-38
CSB	-007	Copy B sign	2-45
DIV	10	Divide	2-16
EAB	21.03	(EA) to EB	2-86
EAD	45	EAU add	2-98
EBA	21.02	(EB) to EA	2-85
EDP	21.12	EAU D.P. mode	2-81
EDV	50	EAU Divide	2-100
EFP	21.14	EAU F. P. mode	2-79
EFU	21.11	Un-normalize F.P.	2-82
EIA	21.01	(EA) to EB, (EB) to EA	2-84
ELL	54	Load LSH of (EA)	2-93
ELN	51	Load ($\overline{M}$) in EA	2-95
ELO	52	Load (M) in EA	2-99
EMU	47	EAU Multiply	2-96
ENO	21.00	EAU Normalize	2-83
EPS	21.06	Skip if (EA) pos.	2-89
ESL	55	Store LSH of (EA)	2-92
ESN	21.07	Skip if (EA) neg.	2-90
ESO	21.10	Skip on EAU O'FLOW	2-91
ESP	21.13	EAU S.P. mode	2-80
ESR	21.04	Skip if EAU not ready	2-87
EST	53	Store (EA)	2-94
ESU	46	EAU Subtract	2-97
ESZ	21.05	Skip if (EA)=0	2-88
EXU	16	Execute (M)	2-57
FLA	-013	Full Left Shift, Arith.	2-52
FLL	-017	Full Left Shift, Log.	2-53
FRA	-012	Full Right Shift, Arith.	2-49
FRL	-014	Full Left Rotate, Log.	2-54

MNE-MONIC	OP CODE	FUNCTION	PAGE
HLT	-000	Halt	2-58
IAB	-006	(A) to (B); (B) to (A)	2-44
IAM	44	(A) to (M); (M) to (A)	2-9
IIB	34	(X)+1; Branch if pos.	2-27
IMS	14	(M)+1; Branch if 0	2-30
LAA	01	(M) to A	2-2
LBA	02	(M) to B	2-3
LCS	57	Switches to A	2-8
LIX	32	(M) to X	2-6
LSA	-011	Left Shift A, Arith.	2-50
LSL	-016	Left Shift A, Log.	2-51
MAA	27	(M) AND (A)	2-34
MEA	26	(M) Exclusive OR (A)	2-33
MCA	37	(M) to Chan. A	2-74
MCB	40	(M) to Chan. B	2-75
MIC	17.24	(Chan.) to M	2-68
MOA	30	(M) OR (A)	2-35
MOC	17.23	(M) to Chan.	2-67
MPY	07	Multiply	2-15
NEG	56	2's comp. (A)	2-37
NOP	-022	No operation	2-59
NOR	-032	Normalize (A) and (B)	2-55
PID	43.0	P.I. Disable	2-60
PIE	43.1	P.I. Enable	2-61
PIR	36	P.I. Return	2-21
RNA	60	Round (A) by (B)	2-17
RSA	-010	Right shift A, Arith.	2-47
RSL	-015	Right shift A, Log.	2-48
SAS	-021	Skip on A sign	2-28
SCN	17.20	Select chan.	2-63
SMA	06	Subtract (M) from (A)	2-14
SMP	35	Skip if (M) pos.	2-29
SNS	13.4	Skip No signal	2-31
SPB	12	Store Place & Branch	2-20
STA	03	(A) to M	2-4
STB	04	(B) to M	2-5
STI	33	(X) to M	2-7
SUN	17.0	Select Unit	2-64
TAB	-005	(A) to B	2-43
TBA	-004	(B) to A	2-42
TBI	-002	(B) to X	2-40
TEX	13.6	Ext. Test	2-73
TCN	13.2	Text Channel	2-72
TUN	13.0	Test Unit	2-71

SEL®

SYSTEMS ENGINEERING LABORATORIES, INCORPORATED

P. O. BOX 9148 / FORT LAUDERDALE, FLA. 33310 / AREA CODE 305 / 587-2900